AFRL-AFOSR-VA-TR-2016-0207

Data-Adaptable Modeling and Optimization for Runtime Adaptable Systems

Roman Lysecky
ARIZONA UNIV BOARD OF REGENTS TUCSON

06/08/2016
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTA2
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
05/31/2016	Final Report	05/01/2015 - 04/30/2016

4. TITLE AND SUBTITLE

Data-Adaptable Modeling and Optimization for Runtime Adaptable Systems

5a. CONTRACT NUMBER

5b. GRANT NUMBER

FA9550-15-1-0143

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Roman Lysecky, Jonathan Sprinkle
University of Arizona
1230 E Speedway Blvd.
(520) 621-6192

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

UNIVERSITY OF ARIZONA
888 N EUCLID AVE
TUCSON AZ 85719-4824

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Office of Scientific Research
875 North Randolph Street, Rm 3112
Arlington VA 22203-1954

10. SPONSOR/MONITOR'S ACRONYM(S)

AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Dynamic data driven application systems (DDDAS) involve complex sensing and decision-making algorithms that operate on vast data streams with dynamic characteristics. As the availability and quality of the sensed data changes, the underlying models and decision algorithms should continually adapt in order to meet desired high-level requirements. Due to the complexity of such dynamic data-driven systems, traditional design time techniques are incapable of producing a solution that remains optimal in the face of dynamically changing data, algorithms, and even availability of computational resources. Additionally, modern approaches to DDDAS design the adaptation laws for dynamic behavior as part of the system itself, thereby resulting in a point solution for that specific application. This research project developed generalized approaches to DDDAS so that the benefits of adaptability can be extended to other applications, without resorting to application-specific point solutions.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Roman Lysecky
U	U	U	SAR	25	19b. TELEPHONE NUMBER *(Include area code)* 520-621-6192

Reset

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18
Adobe Professional 7.0

INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/ monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.

Final Report: FA9550-15-1-0143, Data-Adaptable Modeling and Optimization for Runtime Adaptable Systems

Roman Lysecky, Jonathan Sprinkle
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ, USA
rlysecky@ece.arizona.edu, sprinkle@ece.arizona.edu

1. Abstract

Dynamic data driven application systems (DDDAS) involve complex sensing and decision-making algorithms that operate on vast data streams with dynamic characteristics. As the availability and quality of the sensed data changes, the underlying models and decision algorithms should continually adapt in order to meet desired high-level requirements. Due to the complexity of such dynamic data-driven systems, traditional design time techniques are incapable of producing a solution that remains optimal in the face of dynamically changing data, algorithms, and even availability of computational resources. Additionally, modern approaches to DDDAS design the adaptation laws for dynamic behavior as part of the system itself, thereby resulting in a point solution for that specific application. This research project developed generalized approaches to DDDAS so that the benefits of adaptability can be extended to other applications, without resorting to application-specific point solutions.

2. Summary of Scientific and Technical Accomplishments

In this project we developed a initial Data-Adaptable System Modeling (DASM) theory and design methods that decouple the dynamic adaptation of the system from its runtime implementation. This decoupling simplifies the design of DDDAS applications, while providing a robust runtime framework that automatically infers the dynamic adaptation rules from the application models and optimizes according to the dynamic composition of constraints. We demonstrate each facet of the DASM modeling and optimization process via a video-based vehicle tracking and collision avoidance application, and show how such an approach results in efficient design space exploration when selecting the optimal set of algorithm modalities. When searching for an application configuration within 1% to 5% of optimal, our model-guided approach can achieve speedups of up to 9.3X versus a standard genetic algorithm and speedups of up to 80X relative to a brute force algorithm.

The following summarizes our primary scientific and technical accomplishments:

1. Developed data-adaptable system modeling (DASM) allowing dynamic composition/adaptation/optimization that:
    a. Enables Adaptability-as-a-service for embedded real-time dataflow systems
    b. Captures semantic and programmatic composability by design

   c. Uses sensing, computing, communication, and application-driven requirement changes to trigger adaptation
   d. Quantifies the optimality of system configurations in dynamic execution scenarios
   e. Enables model-guided optimization algorithms that outperform state-of-the-art
   f. Understands the overhead of system reconfiguration under real-time constraints
   g. Designer specifies application alternatives and gets optimization "for free" due to the use of model-based design
2. Applied DASM to: 1) distributed video-based vehicle detection, and 2) vehicle collision avoidance system
3. Developed a new DASM-guided genetic optimization algorithm for exploring the dynamic execution space of the distributed video-based vehicle detection and classification
   a. Results are new capabilities to efficiently optimize the system execution at runtime
   b. Demonstrated up to 26X faster than standard optimization algorithms, with >20% improvement in system fitness compared to optimized point solutions
   c. Enabled by considering the coupling between data and application structure
4. Developed a model-based fuzzy logic classifier synthesis approach
   a. Enables designers to model competing high-level metrics used for dynamic optimization
   b. Supports specification of fuzzy-based fitness rules that capture the relative importance of each high-level metric in determining overall system fitness
   c. Model transforms that estimate fuzzy classifications for each high-level metric at runtime
   d. Demonstrated improvements of system fitness between 26% and 67%, with closer adherence to target design fitness goals compared to traditional piecewise linear formulations

3. Technical Research Overview and Summary of Results

The following provides a high-level overview of the two primary research thrusts under taken in this project, for which complete details can be found in the referenced papers.

3.1 Model-driven Optimization of Data-Adaptable Embedded Systems [1]

Implementing Dynamic Data-Driven Application Systems (DDDAS) requires new methodologies and tools to support both design and synthesis. In this paper, we introduce a modeling tool, called the Data-Adaptable System Model (DASM), that facilitates design by enabling the designer to: 1) specify both an application's task flow as well as various possible implementations for each task; 2) define data types and data quality requirements associated with

the inputs and outputs of each task; 3) build a data quality estimation framework used to estimate data qualities at the output of the system as well as at the output of intermediate tasks; 4) define a set of processing elements (PE) and associated communication delay models; and 5) utilize an efficient model-guided genetic algorithm-based optimization framework to optimize and/or constrain application metrics at runtime. Additionally, the modeling tool is able to parse the designer-created model in order to generate a sample version of the runtime estimation and optimization framework. This estimation and optimization framework dynamically estimates data qualities and other high-level metrics of interest in order to select an optimal (or near-optimal) configuration of tasks. The modeling and optimization features of DASM are presented via a sample video-based vehicle tracking and collision avoidance application.

In order to demonstrate the advantages of the model-guided approach, we executed the design space exploration routine using both a standard genetic algorithm and the presented model-guided genetic algorithm in five different optimization scenarios for the video-based vehicle tracking and collision avoidance application.

Table 1. Five optimization scenarios, each representing a specific vehicle speed differential.

Scenario	Minimum Speed (mph)	Maximum Speed (mph)	Latency Constraint (s)	Resolution Constraint
1	14	18	0.8467	≥ 352 x 288
2	24	32	0.7481	none
3	42	56	0.6184	none
4	50	66	0.5077	none
5	65	88	0.2348	none

Table 1 summarizes the five optimization scenarios. Since the relative speed of the autonomous vehicle and surrounding traffic are expected to influence the application's end-to-end latency constraint, each scenario encapsulates a specific speed differential encountered in typical driving scenarios. For example, scenario 1 corresponds to a 15 mph school-zone in which the expected minimum, maximum and differential speeds are 14, 18, and 4 mph, respectively. Scenario 5, on the other hand, corresponds to a typical freeway driving scenario in which the expected difference in vehicle speed is 23 mph. Each scenario in Table 1 specifies the corresponding latency and resolution constraints required for responsive and safe collision avoidance.

To determine the quality of a task configuration in distinct driving scenarios, the optimal task configuration for each optimization scenario was evaluated in all other scenarios using the fitness criteria of the model-guided GA and designer-specified constraints. Figure 1 presents the performance of each configuration and scenario pair, where, for example, *config 1* corresponds to the optimal task configuration for scenario 1 and so on. Each plot summarizes the relative performance of the non-optimal configurations in comparison to the optimal configuration for a specific scenario. For scenario 1, Figure 1 shows that both configuration 2 and 3 achieve reduced latencies (i.e., -8.7% and -24.9% respectively) than the optimal configuration, configuration 1, at

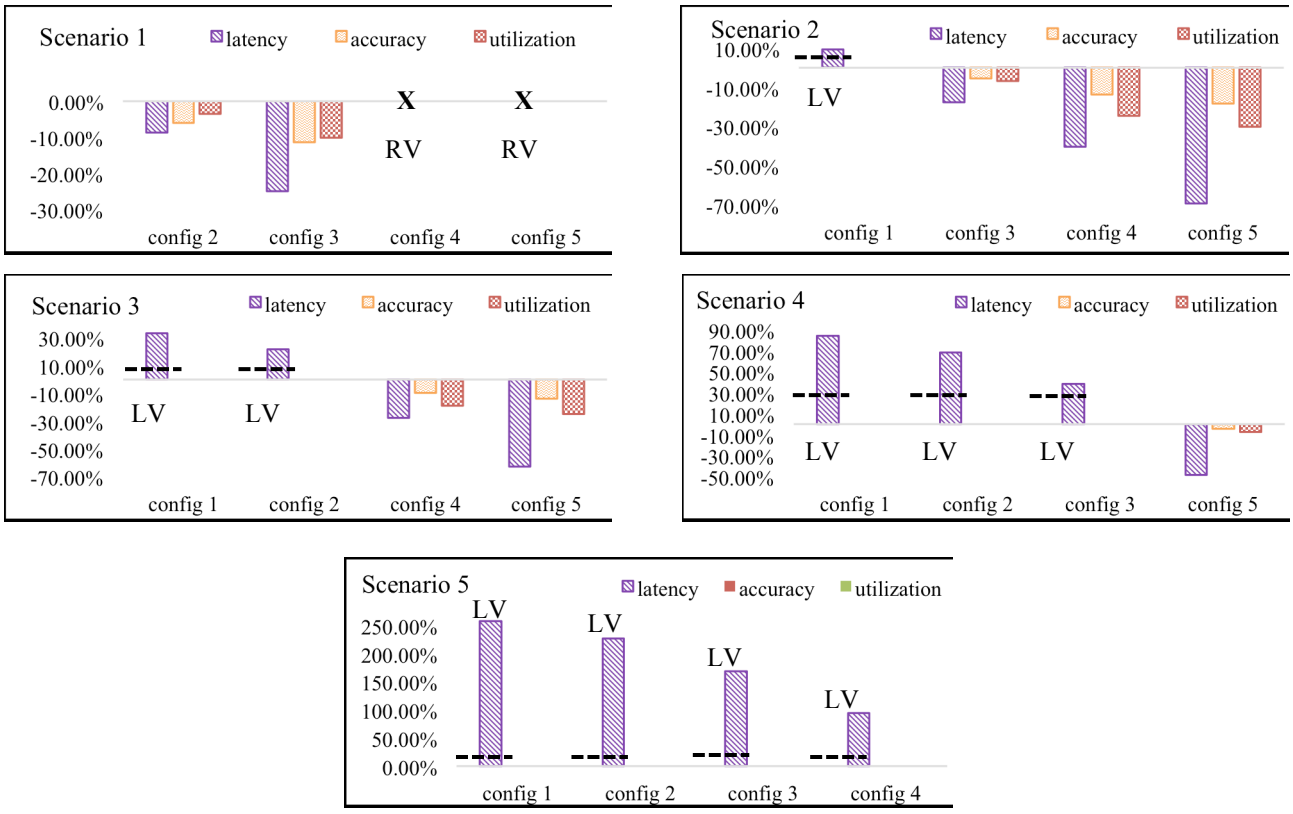

Figure 1. Relative performance of the five selected configurations over the five optimization scenarios. Each graph presents an optimization scenario showing the percent deviation in latency, accuracy, and utilization of each configuration in comparison to the optimal configuration. RV indicates video resolution violation. LV indicates latency constraint violation.

the expense of decreased accuracy. Because the overall application model is configured to optimize accuracy and only consider latency as a constraint that must be met, configurations 2 and 3 are needlessly sacrificing accuracy for improved latency. While configurations 2 and 3 are valid, configurations 4 and 5 are as invalid due to a failure to meet video resolution criteria. In scenario 2, we can see that configuration 1, which is optimized with a more lenient latency constraint of 846.7 ms, is also invalid due to a latency constraint violation (LV).

These results demonstrate that any single optimal configuration can be found to be unacceptably suboptimal, invalid or even unsafe if utilized in other scenarios for which the configuration was not optimized. Therefore, a single static configuration specified at design time is not adequate for DDDAS applications, which are highly susceptible to changing data and environmental characteristics. Instead, such applications require dynamic runtime optimization as specified by DASM.

3.2 Model-based Fuzzy Logic Classifier Synthesis for Optimization of Data-Adaptable Embedded Systems [2]

Dynamic data driven application systems (DDDAS) involve complex sensing and decision-making algorithms that operate on vast data streams with dynamic characteristics. As the

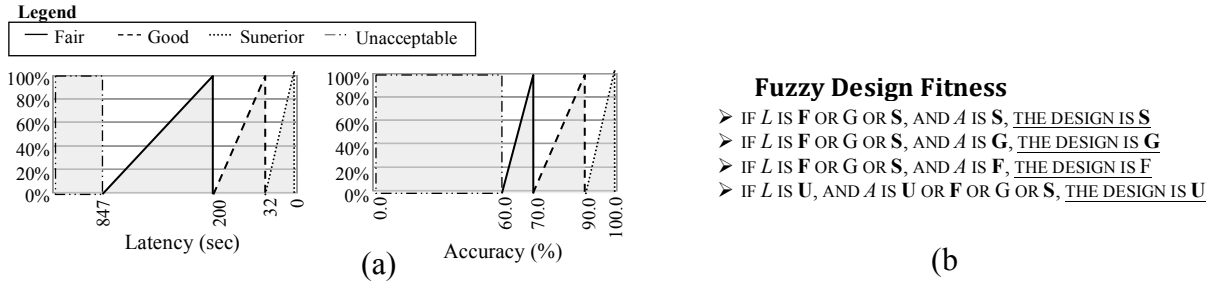

Figure 2. Fuzzy logic-based fitness estimation: (a) classification functions relate DAs to fuzzy classifications; and (b) design fitness rules relate relative importance of fuzzy DA metrics to overall design fitness.

availability and quality of the sensed data changes, the underlying models and decision algorithms should continually adapt in order to meet desired high-level requirements.

To enable both the design and synthesis of runtime DDDAS applications in the face of competing optimization metrics, we developed extensions to the Data-Adaptable System Model (DASM) that enable designers to: 1) specify the competing high-level metrics to be optimized; 2) specify fuzzy-based fitness rules that capture the relative importance of each high-level metric in determining overall system fitness; and 3) model the transforms that estimate fuzzy classifications for each high-level metric at runtime. Using the designer-specified models, DASM generates a prototype version of the runtime estimation and optimization framework that determines an optimal, or near optimal, task implementation configuration given the specified fuzzy logic based optimization criteria.

The DASM modeling and optimization frameworks use a fuzzy logic based formalism to interpret the fitness of individual high-level metrics and the overall system performance. In comparison to other formalisms, such as weighted piecewise linear functions, fuzzy logic allows designers to more intuitively specify tradeoffs between competing metrics and optimization goals. Converting a raw high-level metric value, such as a latency of 100ms, to a fuzzy classification requires the specification of fuzzy metric classification functions. Figure 2(a) presents two example fuzzy metric classification functions for the classification of the latency and accuracy high-level metrics. A fuzzy metric classification function maps a raw metric value to one of four discrete classifications, namely *Unacceptable, Fair, Good,* and *Superior*.

To specify the relative importance of each high-level metric in determining the overall system fitness, designers are tasked with specifying fuzzy design fitness rules within the DASM modeling environment. Fuzzy design fitness rules are English sentences that map the fuzzy classification of each metric to a single fuzzy classification that defines the fitness of the overall system. Figure 2(b) presents four fuzzy design fitness rules. For example, the first rule specifies that if the latency is *Fair*, *Good*, or *Superior*, and the accuracy is *Superior*, then the overall system fitness is *Superior*. Using this approach, the DASM interprets a set of fuzzy metric classifications, such as a latency and accuracy that are 20% *Good* and 50% *Superior* respectively, to an overall design fitness such as 45% *Superior*.

The DASM supports automated runtime synthesis of fuzzy metric classification functions via the specification of a Fuzzy Logic Classification Synthesizer (FLCS) model. The FLCS consists of several Classifier Attribute Transforms (CAT), which transform data and evaluation

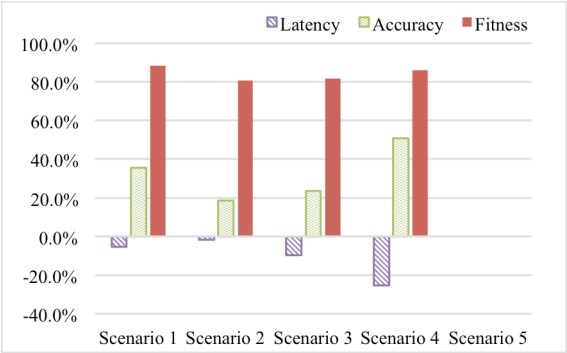

Figure 3. Improvement in metric and overall system fitness due fuzzy logic based optimization in comparison to piecewise weighted linear functions.

attributes into a numerical description of a fuzzy metric classification function. Specifically, a designer is tasked with specifying a CAT for each high-level metric of interest.

To evaluate the benefits of model-guided fuzzy logic classification synthesis for the optimization of DDDAS applications, we executed the prototype runtime optimization framework for the video-based vehicle tracking and collision avoidance application and selected the five dynamic execution scenarios. Each of the five execution scenarios is characterized by the detected speed differential between the autonomous and lead vehicles. We analyzed the benefits of utilizing the presented fuzzy logic based formalisms during DSE by comparing to the results obtained using piecewise linear weighted functions. For each execution scenario, Figure 3 compares the fitness of optimal configurations obtained using fuzzy logic based formalisms to the fitness of optimal configurations obtained using a piecewise weighted linear function. For scenarios 1 through 4, the configurations obtained using fuzzy logic exhibit a moderate degradation in latency fitness in exchange for significant increases in accuracy and overall system fitness. Note that both methods found the same optimal configuration in scenario 5.

On average, the fuzzy based approach produced configurations that sacrificed latency by 8.4%, but showed average improvements in accuracy and overall system fitness of 25.7% and 67.3%, respectively. Importantly, although the fuzzy based approach produced "slower" configurations, these configurations still executed faster than the limiting value of $Latency_{max}$. The fuzzy based configurations still execute within the application's safety constraints, and additionally achieve significant increases in fitness. Thus, it is evident that the fuzzy logic based formalisms more precisely characterize latency as a constraint that must only be met, while accuracy is aggressively optimized.

4. Publications

[1] A. Lizarraga, J. Sprinkle, R. Lysecky. *Model-driven Optimization of Data-Adaptable Embedded Systems*. IEEE Computer Software and Applications Conference (COMPSAC), 2016.

[2] A. Lizarraga, J. Sprinkle, R. Lysecky. *Model-based Fuzzy Logic Classifier Synthesis for Optimization of Data-Adaptable Embedded Systems*. Submitted to ACM SIGBED International Conference on Embedded Software (EMSOFT), *Under Review*, 2016.

Model-driven Optimization of Data-Adaptable Embedded Systems

Adrian Lizarraga, Roman Lysecky, Jonathan Sprinkle
Dept. of Electrical and Computer Engineering
University of Arizona
Tucson, AZ, USA
adrianlm@email.arizona.edu, rlysecky@ece.arizona.edu, sprinkle@ece.arizona.edu

Abstract—**Complex sensing and decision applications such as object tracking and classification, video surveillance, unmanned aerial vehicle flight decisions, and others operate on vast data streams with dynamic characteristics. As the availability and quality of the sensed data changes, the underlying models and decision algorithms should continually adapt in order to meet desired high-level requirements. Due to the complexity of such dynamic data-driven systems, traditional design time techniques are often incapable of producing a solution that remains optimal in the face of dynamically changing data, algorithms, and even availability of computational resources. To assist developers of these systems, we present a modeling and optimization methodology that enables developers to capture application task flows and data sources, define associated quality metrics with data types, specify each algorithm's data and quality requirements, and define a data quality estimation framework to optimize the application at runtime. We demonstrate each facet of the modeling and optimization process via a video-based vehicle tracking and collision avoidance application, and show how such an approach results in efficient design space exploration when selecting the optimal set of algorithm modalities. When searching for an application configuration within 1% to 5% of optimal, our model-guided approach can achieve speedups of up to 9.3X versus a standard genetic algorithm and speedups of up to 80X relative to a brute force algorithm.**

Keywords—Software modeling; dynamic data-driven systems; dynamic optimization; design space exploration

I. INTRODUCTION

Distributed embedded systems are used for a wide variety of sensing and decision applications characterized by their consumption of vast data streams in formats such as video, audio, sensor readings, etc. Examples of such applications include video tracking and surveillance [6][24][10], flight planning of self-aware unmanned aerial vehicles (UAV) [1][15], structural health monitoring [4], and others. The performance and operation modalities of these dynamic data-driven applications are largely dependent on the availability and, importantly, the quality of the incoming data. As these aspects of the data change at runtime, the underlying application should adapt its configuration by implementing more suitable data processing algorithms, or configuring data sensing devices in order to meet designer-specified performance and quality constraints. For example, a distributed video tracking/detection application may need to dynamically switch to a more aggressive video filtering algorithm in response to a detected decrease in the video data's signal-to-noise (SNR) ratio in order to meet detection accuracy constraints.

Implementing data-adaptable embedded systems (DAES) requires new methodologies and tools to support both design and synthesis. In this paper, we introduce a modeling tool, called the Data-Adaptable System Model (DASM), that facilitates design by enabling the designer to: 1) specify both an application's task flow as well as various possible implementations for each task; 2) define data types and data quality requirements associated with the inputs and outputs of each task; 3) build a data quality estimation framework used to estimate data qualities at the output of the system as well as at the output of intermediate tasks; 4) define a set of processing elements (PE) and associated communication delay models; and 5) utilize an efficient model-guided genetic algorithm-based optimization framework to optimize and/or constrain application metrics at runtime.

Additionally, the modeling tool is able to parse the designer-created model in order to generate a sample version of the runtime estimation and optimization framework. This estimation and optimization framework dynamically estimates data qualities and other high-level metrics of interest in order to select an optimal (or near-optimal) configuration of tasks. The modeling and optimization features of DASM are presented via a sample video-based vehicle tracking and collision avoidance application.

The paper is organized as follows. Section II describes related work in the modeling and optimization of embedded systems, along with a brief summary of relevant DAES applications. Section III motivates the need for data-adaptable modeling and design. Sections IV and V describe the modeling and optimization aspects of DASM respectively. Section VI presents the experimental setup and results, and Section VII provides concluding remarks with suggestions for future work.

II. RELATED WORKS

Previous research has levied model-based techniques for the optimization of embedded systems. The MILAN [3] framework specializes in the design, simulation, and synthesis of System On Chip (SoC) applications using model-based techniques. MILAN enables designers to model application tasks and computational resources. For each task, the designer can specify alternate specialized implementations (e.g., C, VHDL) for each compatible resource along with a known latency or energy cost associated with the implementation. The design space exploration (DSE) phase uses the aforementioned models in conjunction with a performance and evaluation model to determine a set of suitable task-to-resource mappings that meet performance and structural constraints. Using a human-in-the-loop process, MILAN employs various simulations to verify functionality and refine the latency and energy cost approximations associated with each task mapping.

The Signal Processing Platform (SPP) tool-suite [19] provides a modeling language called Signal Processing Modeling Language (SPML) that is specialized for the design and synthesis of signal processing applications. Similar to MILAN, the SPP tool supports modeling of tasks and computational resources, and searches the design space to find a set of task-to-hardware

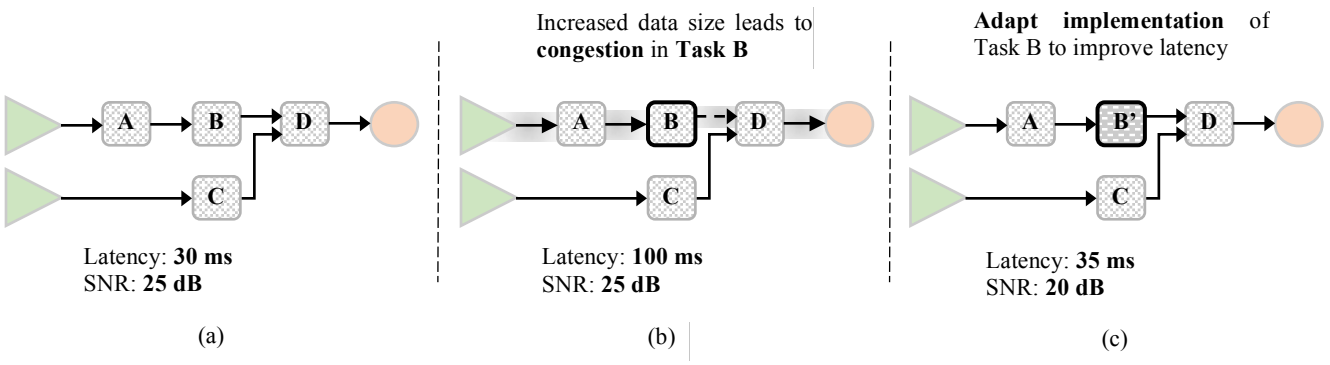

Figure 1. (a) A DAES application achieves an initial latency and output data quality. (b) A change in data size creates congestion between tasks B and D, increasing end-to-end latency. (c) Consequently, a DAES application must be able to adapt task implementations in order to account for data variability and optimize system performance at runtime.

mappings that meet resource and performance constraints. However, the SPML only supports parameterized task models as opposed to alternative implementations. The SPP and MILAN tools are design time tools and thus do not support runtime optimization.

The SPP and MILAN tools produce a set of task mappings based on known, design time, performance characteristics associated with task hardware pairings. DAES applications require additional runtime tools that can further customize task implementations and mappings in response to varying data qualities, dynamic constraints, and changing availability of computational resources. Additionally, designers often need to evaluate system configurations using application-specific metrics in addition to the power and latency measurements employed by SPP and MILAN.

The DARES project [18][29] models data configurability of application tasks supporting the runtime reconfiguration of hardware accelerators implemented with an FPGA. Unlike other approaches, the application model can be synthesized into a physical device in order to enable runtime task allocation in order to optimize performance. However, the DARES approach does not consider the specification or adaptability of the algorithms themselves, only the type of data being processed.

Many data-driven applications, also known as Dynamic Data-Driven Application Systems (DDDAS) [13][17][23][26][31], critically depend on the ability to evaluate the quality of incoming data in order to perform the intended task. For example, Allaire et al. [1] applies data-driven methods to the navigation of self-aware aerospace vehicles. The automated vehicle must be able to determine the feasibility of the next required maneuver given the vehicle's current health and flight capabilities. Specifically, the vehicle must employ a prediction task with the highest fidelity given the quality of the sensed data and the amount of time until the maneuver is performed. Failure to select the correct task may result in system failure.

Frew et al. [9] similarly uses this paradigm to optimize the flight path of a UAV, exploiting wind field data in order to determine a flight path that conserves energy, and thus extends mission duration. Wind field data can be extracted at runtime using a variety of algorithms/resources, where the appropriate method depends on the set of available resources and the required fidelity.

Bazilevs et al. [4] employ structural health monitoring sensors to monitor the health of a wing or spinning blade composed of composite materials. This structural health data is imported into a simulation of the wing/blade in order to calculate a stress map. The control software uses the computed stress data to determine metrics of interest, such as maximum blade deflection, vibration frequency, lifetime, and others. The control software then searches the design space to find a suitable mode of operation for the physical structure.

Autonomous vehicles comprise complex hardware and software subsystems that must interact appropriately to achieve functional, performance, and safety goals. Lane detection subsystems, such as those developed in [14] and [32], use sophisticated modeling and image processing techniques to detect road markings and lane departure. Adaptive cruise control subsystems may employ models for vehicle dynamics [21] or machine-learning techniques [7] to intelligently adapt the vehicle's speed or avoid collisions [11][12][2]. Importantly, these subsystems must perform correctly over a wide range of operating conditions. For example, changes in visibility or road conditions may necessitate the use of more accurate algorithms while operation at higher vehicle speeds may require faster, more reactive algorithms to meet safety constraints.

Importantly, these data-driven applications use dynamic characteristics of the input and output data, along with changes in operating conditions, to dynamically reconfigure the system's algorithms, operating modes, or hardware. Dynamic changes in input/output data typically occur on the order of seconds to minutes, and in response to these changes, the runtime reconfiguration process must complete within milliseconds to seconds.

III. Data-Adaptable Approach

Designing data-adaptable systems necessitates incorporating system adaptability at multiple levels. First, the system must be able to handle dynamic availability of data inputs. For example, a vehicle tracking application that utilizes both video and GPS data to track vehicles of interest may often encounter situations in which it is unable to retrieve video or GPS data in remote areas. A data-adaptable approach should enable such an application to adequately deal with a degradation, or complete loss, of data without unnecessary interruptions.

Second, a data-adaptable application must account for the availability of computing resources in the case of distributed embedded applications. For example, an autonomous vehicle system should ideally continue to function in the event that one or more processing elements malfunction or become unavailable. Specifically, the underlying task implementations should adapt to ensure continued operation at the required level of fidelity based on the current set of available computing resources. Note that although the DASM tool supports resource modeling that enables re-optimization in response to changes with resource availability,

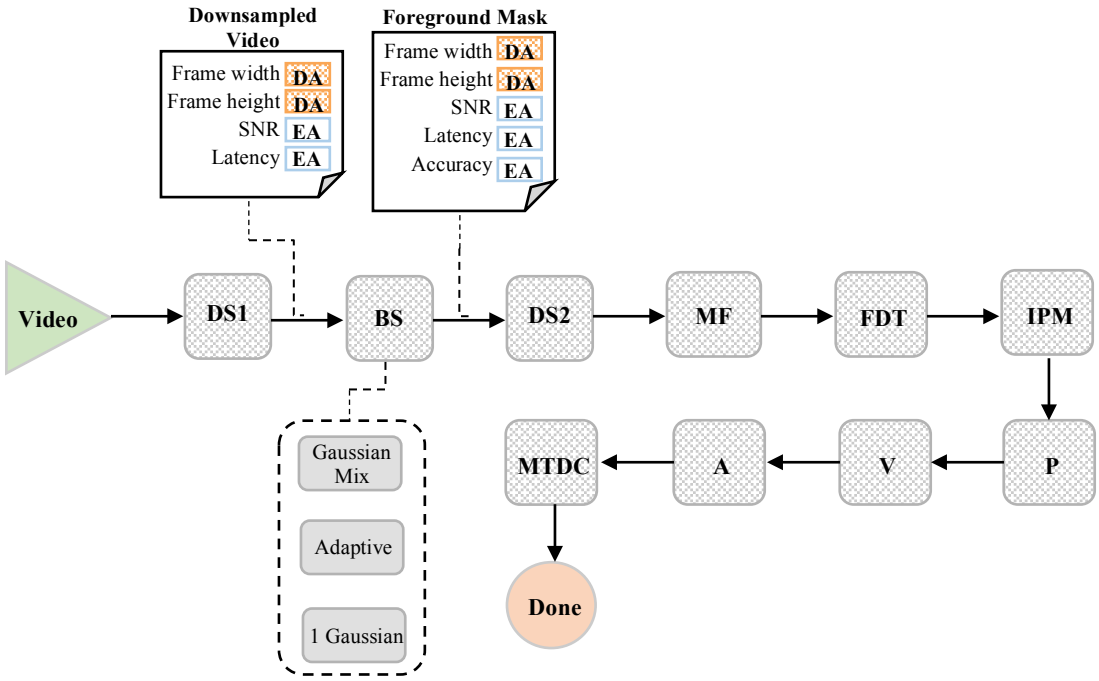

Figure 2. Task flow modeling for a video-based vehicle tracking and collision avoidance application. Modeling a DAES application requires specifying the end-to-end task flow, data types, and available task options.

this paper does not consider dynamically changing computational resources.

Fig. 1(a) presents an example task flow of a generic DAES with two data sources and four tasks. Triangles in the diagram represent the data sources (e.g., camera, and GPS receiver), rounded rectangles represent the application tasks, and a circle denotes the application's final output. The arrows represent the data consumed and produced by each task. This pedagogical application initially produces an output with a latency of 30 milliseconds and a SNR of 25 dB. A change in input data size, as depicted in Fig. 1(b), leads to congestion in communication between Tasks B and D, resulting in an increased end-to-end latency of 100 ms. A single static task implementation exhibits different performance characteristics for varying execution scenarios and data qualities, which may result in an unacceptable degradation in performance, as shown in Fig. 1(b), or constraint violations that could lead to a system failure. The system in Fig. 1(c) compensates for the decrease in latency by adapting the implementation of Task B to one that can adequately process the new incoming data.

As illustrated by this example, designing data-adaptable systems requires a detailed understanding of the end-to-end application. That is, a designer must understand the variability in input data characteristics, and how this variability impacts a task's or even the entire application's performance metrics. Additionally, the designer must be able to apply this knowledge to produce a set of distinct task implementations that can be interchanged in order to adapt to the aforementioned variability.

The complexities involved in designing data-adaptable systems in this fashion necessitate the development of new methodologies and tools that elevate design to a higher level of abstraction. For this reason, we introduce a modeling framework for designing data-adaptable systems that allows designers to model tasks, input and output data types,

IV. Data-Adaptable System Modeling

We propose a data-adaptable system model (DASM), implemented as a domain-specific model within the Generic Modeling Environment (GME) [30]. Our DASM is a high-level design model that enables the specification and optimization of DAES using a domain-specific modeling language. System specification entails modeling all tasks, specifying alternate task implementations for those tasks, and specifying the set of available PEs. Optimization is the process of exploring the design space to select a unique implementation for each task, along with a task to PE mapping and schedule, so that designer performance constraints are met or optimized.

Fig. 2 presents an example model for a video-based vehicle tracking and collision avoidance application, which is utilized throughout this paper to illustrate the proposed modeling and optimization methodology. This application analyzes video images to detect lead vehicles, track their location, and determine the minimum traveling distance (MTD) necessary to match the lead vehicle's speed and avoid collision. The MTD is compared to the current distance to the lead vehicle to determine the necessary deceleration value.

Several object detection and tracking implementations in the literature [6][24][10] explore specific task flows that vary from each other depending on the tracking object and video characteristics. Given the potential variability in video data characteristics (e.g., frame size, SNR) and the wide variety of possible implementations for common detection tasks [24], such an application is a suitable case study for the DASM methodology.

A. Tasks and Data Types

A DASM consists of a set of connected tasks that define the task flow of an application. *Tasks* are depicted as rounded rectangles and represent abstract processing routines that consume and produce data types. For example, the object detection and

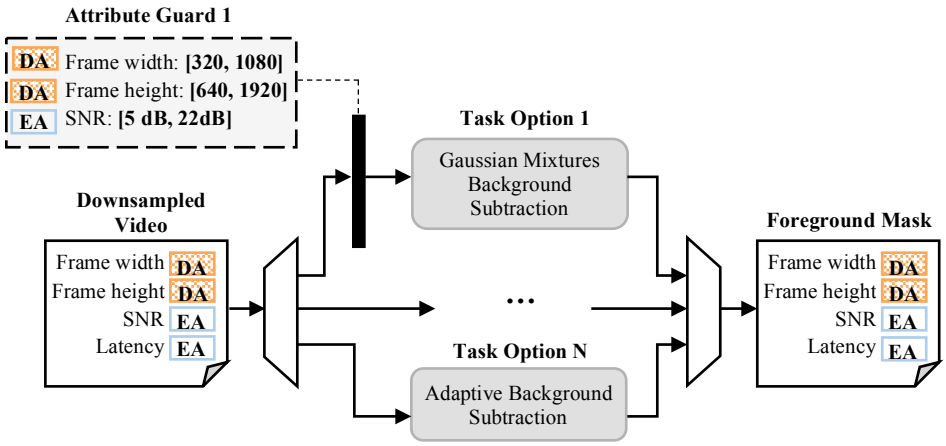

Figure 3. Each Task contains Task Options that transform input data to output data. Task Options optionally specify attribute guards to indicate the acceptable values for each input attribute.

tracking model in Fig. 2 contains a sequence of ten tasks: Video Capture, Downsampling (DS1), Background Subtraction (BS), Downsampling (DS2), Morphological Filter (MF), Feature Detection/Tracking (FDT), Inverse Perspective Mapping (IPM), Position Estimation (P), Velocity Estimation (V), Acceleration Estimation (A), and Minimum Travel Distance Calculation (MTDC).

A *Data Type* is a high-level type specifying a collection of individual data items called *Data Attributes*. A data type is depicted as a rectangle with a folded corner. Data types represent atomic units, or tokens, that are transferred between all tasks in the model. Fig. 2 describes two of the ten data types in the video-based vehicle tracking and collision avoidance model: *Downsampled Video*, and *Foreground Mask*.

Attributes describe properties of their parent data types. The values of a data type's attributes are set by the task implementation producing the data type. We distinguish between two types of attributes: *Data Attributes* (DA) and *Evaluation Attributes* (EA). Data attributes are properties that describe a data type in an actual implementation. For example, all video frames can be associated with width and height attributes to specify the frame's size. On the other hand, evaluation attributes are a product of the modeling framework and are used to evaluate task options for the purposes of task flow optimization, such as latency, accuracy, etc. The latency EA accumulates the execution latency for each task in order to estimate a final application latency. Accuracy is a metric that relates the performance (i.e., precision and recall) of algorithms like background subtraction or feature matching.

Although this application could potentially be decomposed into tasks of finer granularity, we chose a subset that adequately describes the application's most notable components. The *Downsampling* task optionally lowers the input frame's resolution for downstream compatibility. The *Background Subtraction* task distinguishes foreground objects from the background in order to create a foreground mask. The *Morphological Filtering* task removes small, or noisy, foreground objects and uses connected-component labeling to isolate individual objects in the foreground mask. The *Feature Detection* task uses keypoint detectors to match objects between frames. The *Inverse Perspective Mapping* task estimates the lead vehicle's distance using known camera and scene parameters, and the *Position, Velocity*, and *Acceleration* tasks use numerical integration to estimate the lead vehicle's kinematics with a certain degree of accuracy. Finally, the *Minimum Traveling Distance Calculation* task calculates the MTD and compares it to the lead vehicle's current distance to avoid collision.

B. Task Options and Attribute Guards

A task may specify one or more implementations, known as *Task Options*. For example, the background subtraction task in Fig. 2 specifies three distinct task options: Gaussian Mixture, Adaptive, and Single Gaussian. All task options for a particular task must share the same input and output data types in order to ensure data type compatibility with neighboring tasks.

Because each task option represents a specialized implementation of a task, the designer may optionally choose to impose unique input data requirements for each task option using a *Data Attribute Guard*. Fig. 3 shows the available task options for the Background Subtraction task along with pertinent data attribute guards, which are visually represented as black vertical bars. Data attribute guards allow designers to specify the range of data attribute values for which the task option is compatible.

A task option's attribute guard may represent a restriction in functional compatibility such that the task option is known not to work correctly for input data attribute values outside the specified range. Alternatively, designers may choose to employ domain knowledge in specifying attribute guards to limit the execution of a task option to input DA values for which that task option is more suitable than other implementations. For example, the attribute guard in Fig. 3 for the Gaussian mixtures task option specifies that input video frames should have a SNR in the range of 5 dB to 22 dB. While the Gaussian Mixture task option can theoretically execute for any values of SNR, prior design experience indicates that the Gaussian Mixture task option only shows significant performance (e.g., latency and F-measure) gains over other less complex implementations (e.g., adaptive background subtraction) with noisier data [10][5][22].

C. Data and Evaluation Attribute Transforms

A task option is an abstract representation of a particular algorithm or routine. From the model's perspective, a task option simply transforms an input data type (e.g., Video Frame) to an output data type (e.g., Downsampled Frame). A designer specifies the semantics behind this transformation by using attribute transforms. As shown in Fig. 4, a *Data Attribute Transform* (DAT) takes in one or more input data attributes and determines the value of one, or more, data attributes belonging to the output data type.

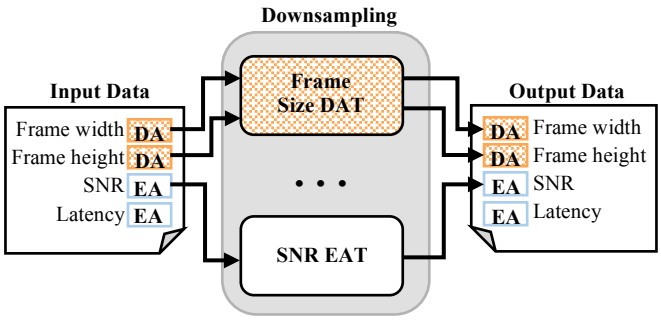

Figure 4. A Task Option consists of Data Attribute Transforms and Evaluation Attribute Transforms.

Similarly, an *Evaluation Attribute Transform* (EAT) determines the value of one, or more, evaluation attributes based on input evaluation attributes.

Attribute transforms are not necessarily implementations of the actual task option algorithm, but instead are more likely simple routines or mathematical expressions that quickly estimate the value of several attributes. For example, utilizing the results in [8] and [16], the output SNR, in decibels, due to downsampling at a ratio of x can be roughly estimated as:

$$SNR_{OUT} = SNR_{IN} + 20\log(\sqrt{x}) \quad (1)$$

Likewise, the accuracy evaluation attribute can be estimated using a probabilistic model [25], or by fitting an equation to empirical data from [27], as was done for this model.

D. Computational Resource and Communication Delay Modeling

As the latency of a task is inexorably linked to the characteristics of the underlying processing hardware, DASM supports the specification of computing devices, which consist of processing elements and memory models, to enable the mapping of tasks to various processing elements. Fig. 5 shows a sample model consisting of two devices, each with two processing elements and a memory.

The latency EATs specified for each task option correspond to a specific base computing device. To determine the latency effects of mapping a task to a different device, DASM calculates a set of scaling parameters that relate the speedup or slow-down associated with the new mapping. These scaling parameters are calculated from designer-specified hardware parameters (e.g., frequency, average memory delay) and task benchmarking results (e.g., average CPI, average miss rate.).

As shown in Fig. 5, DASM also employs designer-specified inter-PE and intra-PE communication models that approximate and abstract the communication delay between devices (e.g., Ethernet communication) and the communication delay between processing elements (e.g., bus and shared memory transactions) within the same device. The designer specifies these communication models in a similar manner as an EAT, that is, as an approximate algorithm that utilizes the size of input data to determine the expected latency.

V. Model-Guided Optimization

A completed DASM encodes all possible task options for each of the application's main tasks and defines how each task option affects the quality of the application's output and intermediate data types. The optimization procedure evaluates each task option in order to find a task option configuration and resource mapping that both optimizes a certain data type attribute and satisfies any number of constraints.

The optimization framework allows a designer to mark any data attribute within the model as the sole optimization variable or as a constraint variable that must exceed or fall below a specified threshold value. For example, the application model presented in Fig. 2 can assign the final accuracy and latency data type attributes as the optimization and constraint variables respectively. Therefore, the design space exploration will explore the design space to find a task option configuration that yields the optimal accuracy while ensuring that the final application latency is within a specified range (e.g., less than 400 ms).

A. Design Space Exploration: Model-guided Genetic Algorithm

With the DAES approach, a novel model-guided genetic algorithm is employed to explore the design space and optimize

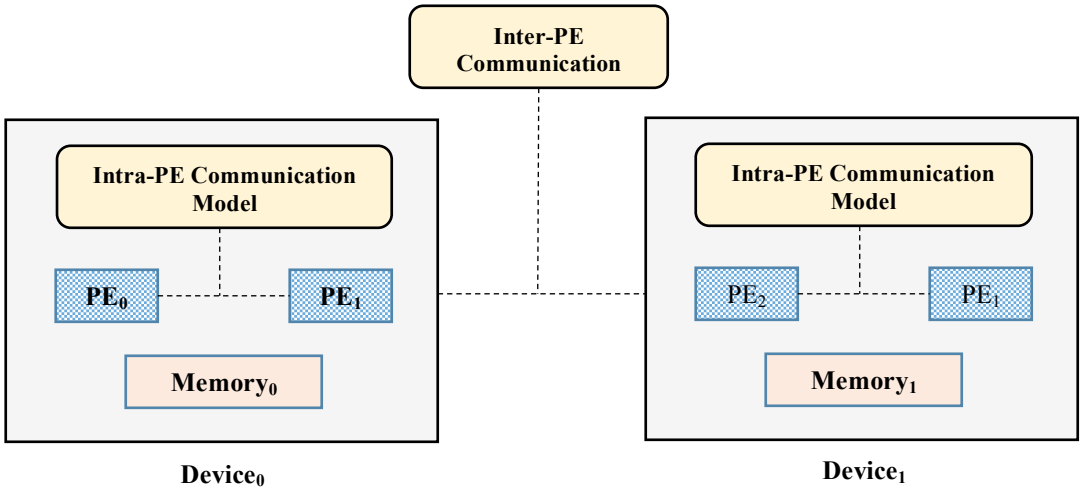

Figure 5. Example of a designer-specified Resource and Communication model within DASM showing two devices, each with two processing elements, a memory and intra-PE communication models. The inter-PE communication block models communication between PEs on separate devices.

the system implementation. Genetic algorithms, which explore the design space by combining and mutating portions of candidate solutions, are well suited for problems in which each candidate solution has a genetic representation. That is, each candidate solution can be represented as a set of properties, in which each property can be assigned one of several possible values.

A candidate solution for a DASM is a complete task option and resource mapping assignment with the genetic representation:

$$Config = < T_1, T_2, \ldots, T_N, M_1, M_2, M_N > \quad (2)$$

where T_N is a configurable variable for task N whose possible values include the task options defined for that particular task, and M_N is a configuration variable that represents the mapping of task N to particular PE.

A standard genetic algorithm [28] functions by (1) generating a random initial population of candidate solutions, (2) selecting parent solutions for crossover with a probability proportional to their fitness, (3) crossing-over parents to generate a child solution, (4) randomly mutating the child to promote variability in solutions, and (5) repeating. Our standard genetic algorithm, which is used as a baseline for comparison to a more sophisticated model-guided approach, operates in a similar manner. However, the population generation stage repeats until all members of the initial population are valid.

A model-guided genetic algorithm, on the other hand, can leverage information from the DASM to enhance several aspects of the genetic algorithm. Generating an initial population by randomly selecting task options frequently results in invalid task option configurations due to the compatibility constraints defined by the attribute guards. For example, randomly selecting a task option in an upstream task could result in output data that is incompatible with the attribute guard of the downstream task. However, by utilizing the compatibility information defined in the model's attribute guards during population generation, the model-guided genetic algorithm can generate an initial population in which every individual is guaranteed to meet the aforementioned data to task compatibility requirements.

The model-guided initial population generation (MGPG) algorithm traverses the task flow graph in reverse order (i.e., from the end to the input data sources) using depth-first search (DFS). At each task T, the algorithm determines the compatibility of each task option T_0 with the input data by inspecting the attribute guards. If a task option is compatible, the task option is inserted into a valid task option list. Once all valid task options for a task node have been considered, the algorithm randomly picks one task option for that task node. Each complete search of the graph yields one individual for the initial population. Therefore, the algorithm is repeated until enough task option configurations have been generated.

A model-guided crossover (MGC) algorithm is also informed by the attribute guards within the DASM to ensure generated offspring are valid. First, the algorithm creates a list containing all possible crossover points. Next, the algorithm picks a random crossover point from the list, generates a child using this crossover point, and returns the child if it is valid. Otherwise, the algorithm removes the crossover point from the list and repeats the process until a valid child is generated or the list becomes empty.

The mutation step is critical for promoting population diversity and thus avoiding local minima. The implemented genetic algorithms use a mutation probability of 3%, which corresponds to the smallest mutation rate necessary to find the optimal task option configuration when using the standard genetic algorithm over a maximum of 500 generations. If a mutation operation does not result in a valid (i.e., compatible) task option configuration, the model-guided mutation (MGM) algorithm does not perform the mutation, whereas the standard genetic algorithm implementation performs the mutations and sets the configuration's fitness to zero.

Lastly, the modeling framework enables the designer to specify the genetic algorithm's fitness function as an additional EAT that transforms the optimization attributes into a fitness score using simple mathematical equations. For our current application, we consider the following fitness function for

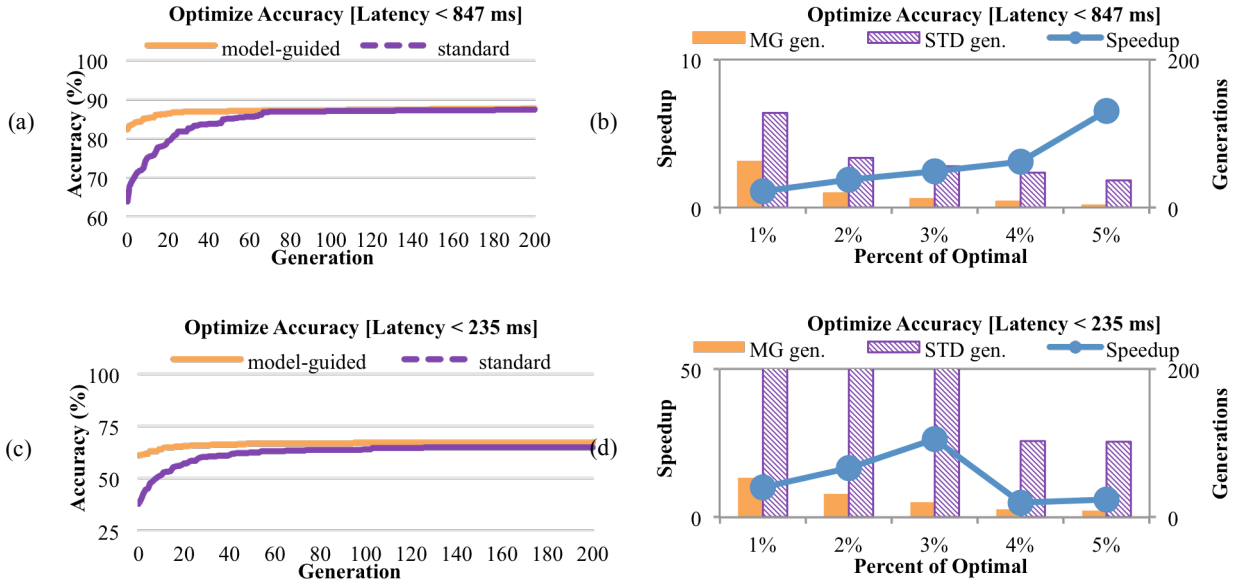

Figure 6. (a) Accuracy score per each generation of the model-guided and standard optimization genetic algorithms and (b) execution speedup for optimization scenario 1. Similar data is presented for optimization scenario 5 in charts (c) and (d).

accuracy and a constraint for latency.

$$fitness = Accuracy_{MTDC} \quad (3)$$

$$Latency_{SCHED} < Latency_{CONSTRAINT} \quad (4)$$

The fitness function for optimizing accuracy directly uses the accuracy value from the last task, which approximates the accuracy of the MTD and is already a number between 0 and 100.

Note that $Latency_{SCHED}$ is the overall latency of the application considering the latency accrued from each task's latency EAT after mapping, the latency accrued from communication delay modeling, and determination of a static task schedule on potentially parallel PEs. In fact the $Latency_{SCHED}$ value is actually the period of the task schedule, which is derived using a custom synchronous data flow (SDF) scheduling algorithm for multiprocessors. For simplicity, the task scheduling algorithm assumes that tasks cannot be interrupted until completion.

VI. Experiments and Results

In order to demonstrate the advantages of the model-guided approach, we executed the design space exploration routine using both a standard genetic algorithm and the presented model-guided genetic algorithm in five different optimization scenarios for the video-based vehicle tracking and collision avoidance application.

TABLE 1. Five optimization scenarios, each representing a specific vehicle speed differential.

Scenario	Minimum Speed (mph)	Maximum Speed (mph)	Latency Constraint (s)	Resolution Constraint
1	14	18	0.8467	≥ 352 x 288
2	24	32	0.7481	none
3	42	56	0.6184	none
4	50	66	0.5077	none
5	65	88	0.2348	none

Table 1 summarizes the five optimization scenarios. Since the relative speed of the autonomous vehicle and surrounding traffic are expected to influence the application's end-to-end latency constraint, each scenario encapsulates a specific speed differential encountered in typical driving scenarios. For example, scenario 1 corresponds to a 15 mph school-zone in which the expected minimum, maximum and differential speeds are 14, 18, and 4 mph, respectively. Scenario 5, on the other hand, corresponds to a typical freeway driving scenario in which the expected difference in vehicle speed is 23 mph. Each scenario in Table 1 specifies the corresponding latency and resolution constraints required for responsive and safe collision avoidance.

Fig. 6 compares the performance of the model-guided and standard genetic algorithms over optimization scenarios 1 and 5. Each optimization scenario in Fig. 6 presents two plots: plots (a) and (b) correspond to scenario 1 and plots (b) and (c) correspond to scenario 5. The plots on the left presents the best fitness score (averaged over 100 runs) encountered during each generation of both genetic algorithms. The plot on the right contains (i) a bar graph depicting the number of generations (averaged over 100 runs) required to find a solution within a certain percentage of the optimal solution, and (ii) a trend line depicting the speedup of the model-guided genetic algorithm over its standard genetic algorithm counterpart on a 4GHz Intel i5 processor.

Fig. 6 (a) and (c) show that using a model-guided GA results in a better initial population with an improved average fitness score of 18.5% and 23.3% respectively. Additionally, the fitness of the solutions produced by the standard GA in scenario 5 never converges to that of the model-guided GA. This is due to the fact that less task option configurations are valid in the design space as the latency constraint becomes more demanding.

The bar graph in Fig. 6 (b) shows that the model-guided GA can reach a valid configuration with a fitness score within 1% to 5% of the optimal with speedups ranging from 1.2X to 6.5X respectively. Thus, although the model-guided GA finds all solutions faster, it achieves even greater performance gains when searching for a configuration farther from the optimal. However, the speedups presented in Fig. 6 (d) are not increasing with the required percent of optimality only because the standard GA failed to find configurations within 1% to 3% of optimal until approximately the 1000th generation, thus increasing the value of the first three speedup data points up to 26X.

Fig. 7 presents the execution time for the model-guided GA in scenarios 1 and 5. For scenario 1, the model-guided GA's runtime ranges from 1695 ms (62 generations) at 1% of optimal to 82 ms (3 generations) at 5% of optimal. Similarly, the runtime for scenario 5 ranges from 1449 ms (53 generations) at 1% of optimal to 246 ms (9 generations) at 5% of optimal. Compared to a brute force optimization algorithm, which explores all 3402 possible task option configurations taking 5.7 seconds, the model-guided approach achieves a speedup of up to 544X. In that case, the model-guided GA was able to find the target task option configuration in a single generation. The model-guided approach's lowest observed speedup in comparison to the exhaustive algorithm was 8.8X, in which case the model-guided GA was able to find a solution within 1% of optimal in 62 generations.

Notably, the model-guided approach achieves greater speedups when the optimization requirement is relaxed, i.e., as the percent of optimal increases, making a configuration which is around 5% of optimal a suitable design space exploration target.

The model-guided genetic algorithm is composed of smaller model-guided operations, namely MGPG, MGC, and MGM. Over the five optimization scenarios, each executed 100 times, the

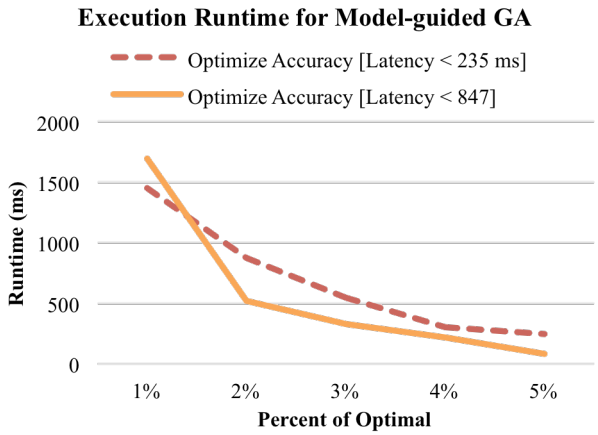

Figure 7. Execution runtime of the model-guided GA for the optimization scenarios 1 and 5.

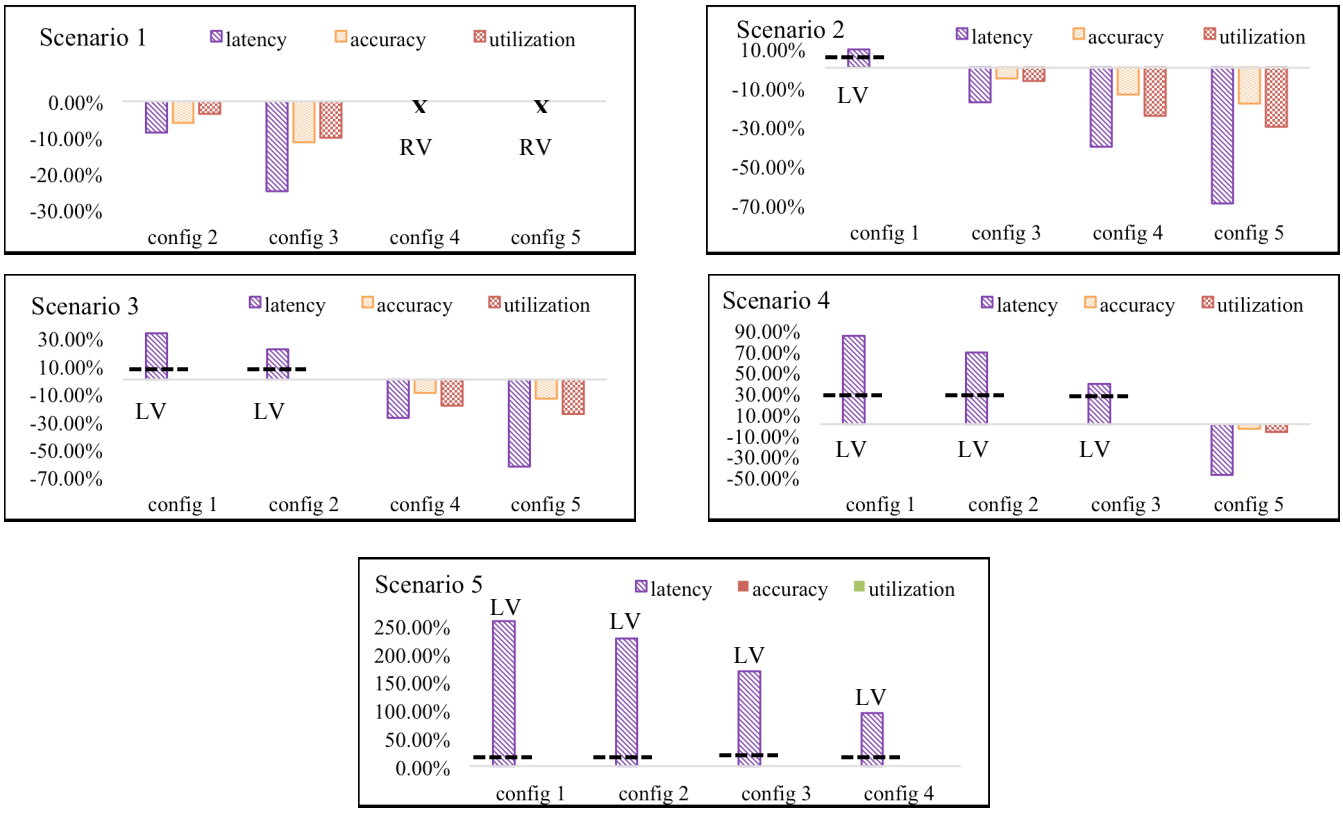

Figure 8. Relative performance of the five selected configurations over the five optimization scenarios. Each graph presents an optimization scenario showing the percent deviation in latency, accuracy, and utilization of each configuration in comparison to the optimal configuration. Video resolution violations are indicated with the acronym RV, and latency constraints violations are indicated with an LV.

model-guided initial population generation was found to account for a majority of the performance improvement of the model-guided GA. Specifically, MGPG accounts for an average improvement of 76% over a standard GA. The MGC step accounts for a lesser, yet still significant, average improvement of 22% over a standard GA. In contrast, the MGM step within the GA account for only a 1-2% improvement. We note that in applications or design spaces in which a higher mutation rate/probability is required, model-guided mutation may show increased benefits.

To determine the quality of a task configuration in distinct driving scenarios, the optimal task configuration for each optimization scenario was evaluated in all other scenarios using the fitness criteria of the model-guided GA and designer-specified constraints. Fig. 8 presents the performance of each configuration and scenario pair, where, for example, *config 1* corresponds to the optimal task configuration for scenario 1 and so on. Each plot summarizes the relative performance of the non-optimal configurations in comparison to the optimal configuration for a specific scenario. In addition to the percentage increase or decreases in accuracy and latency values, we also report the relative change in PE utilization for each configuration.

For scenario 1, Fig. 8 shows that both configuration 2 and 3 achieve reduced latencies (i.e., -8.7% and -24.9% respectively) than the optimal configuration, configuration 1, at the expense of decreased accuracy. Because the overall application model is configured to optimize accuracy and only consider latency as a constraint that must be met, configurations 2 and 3 are needlessly sacrificing accuracy for improved latency. While configurations 2 and 3 were at least valid, configurations 4 and 5 can be marked as invalid by the designer due to a failure to meet video resolution criteria, i.e., a resolution violation (RV). In this example, configurations 4 and 5 utilize a low resolution that are determined unsafe in the 15 mph school-zone represented by scenario 1.

In scenario 2, we can see that configuration 1, which is optimized with a more lenient latency constraint of 846.7 ms, is also invalid due to a latency constraint violation (LV). In fact two, three, and four of the configurations in scenarios 3, 4 and 5, respectively, exhibit latency violations due to the increasingly stricter latency constraints.

The data in Fig. 8 demonstrates that any single optimal configuration can be found to be unacceptably suboptimal, invalid or even unsafe if utilized in other scenarios for which the configuration was not optimized. Therefore, a single static configuration specified at design time is not adequate for dynamic DAES applications, which are highly susceptible to changing data and environmental characteristics. Instead, such applications require dynamic runtime optimization as specified by DASM.

VII. Conclusions and Future Work

We presented a modeling and optimization methodology that enables the specification and optimization of data-adaptable embedded systems. Specifically, the presented modeling framework allows designers to construct models specifying an application's task flow, task options, and computational resources; the input and output data types for each task; the compatible input

values for each task option; and the mathematical formalism used to transform input data attributes to output data attributes. A model-guided genetic algorithm utilizes information encapsulated in the model to prune the design space of incompatible configurations, which results in both faster and better optimization compared to standard genetic algorithm implementation. Experimental results demonstrate that static configurations may exhibit unacceptable performance degradations under various execution scenarios, resulting in invalid and potentially unsafe configurations. Future work entails expanding the modeling framework to allow the designer to specify a fitness function that can optimize multiple competing metrics, formalisms for effectively defining acceptable runtime tradeoffs between those competing metrics, and implementing a more sophisticated stopping criteria for the genetic algorithm to target configurations within a few percentage points of optimal.

ACKNOWLEDGMENT

This research was supported by the Air Force Office of Scientific Research under Grant No. FA9550-15-1-0143.

REFERENCES

[1] D. Allaire, G. Biros, J. Chambers, O. Ghattas, D. Kordonowy, and K. Willcox, "Dynamic Data Driven Methods for Self-aware Aerospace Vehicles," International Conf. on Computation Science (ICCS), 2012, pp. 1206-1210.

[2] V. L. Bageshwar, W. L. Garrard, and R. Rajamani, "Model Predictive Control of Transitional Maneuvers for Adaptive Cruise Control Vehicles," IEEE Transactions on Vehicular Technology, vol. 53, issue 5, 2004, pp. 1573-1585.

[3] A. Bakshi, V. K. Prasanna, and A. Ledeczi, 2001. "MILAN: A Model Based Integrated Simulation Framework for Design of Embedded Systems," In Proc. of the 2001 ACM SIGPLAN workshop on Optimization of middleware and distributed systems, 2001, pp. 82-93.

[4] Y. Bazilevs, A. L. Marsden, F. Lanza di Scalea, A. Majumdar, and M. Tatineni, 2012. "Toward a Computation Steering Framework for Large-Scale Composite Structures Based on Continually and Dynamically Injected Sensor Data," International Conf. on Computation Science (ICCS), 2012, pp. 1149-1158.

[5] Y. Benezeth, P. Jodoin, B. Emile, H. Laurent, and C. Rosenberger, 2010. "Comparative study of background subtraction algorithms," Journal of Electronic Imaging, Society of Photo-optical Instrumentation Engineers (SPIE), 2010.

[6] N. Chintalacheruvu, and V. Muthukumar, "Video Based Vehicle Detection and Its Application in Intelligent Transportation Systems," Journal of Transportation Technologies, 2012, pp. 305-314.

[7] C. Desjardins, and B. Chaib-draa, "Cooperative Adaptive Cruise Control: A Reinforcement Learning Approach," IEEE Transactions on Intelligent Transportation Systems, vol. 12, issue 4, 2011, pp. 1248-1260.

[8] J. Farrell, M. Okincha, M. Parmar, and B. Wandell, "Using visible SNR (vSNR) to compare the image quality of pixel binning and digital resizing," In Proc. of the SPIE, 2010.

[9] E. Frew, B. Argrow, A. Houston, C. Weiss, and J. Elston, 2013. "An Energy-Aware Airborne Dynamic Data-Driven Application System for Persistent Sampling and Surveillance," International Conf. on Computation Science (ICCS), 2013, pp. 2008-2017.

[10] Y. Gedeoglu, "Moving Object Detection Tracking and Classification for Smart Video Surveillance," Master of Science Thesis, 2004.

[11] F. Jimenez, E. J. Naranjo, and O. Gomez, "Autonomous Manoeuvring Systems for Collision Avoidance on Single Carriageway Roads," Sensors, vol. 12, issue 12, 2012.

[12] N. Kehtarnavaz, N. Groswold, K. Miller, and P. Lascoe, "A Transportable Neural-Network Approach to Autonomous Vehicle Following," IEEE Transactions on Vehicular Technology, vol. 47, issue 2, 1998, pp. 694-702.

[13] A. M. Khaleghi, D. Xu, Z. Wang, M. Li, A. Lobos, J. Liu, and Y. J. Son, "DDDAMS-based Planning and Control Framework for Surveillance and Crowd Control via UAVs and UGVs," Expert Systems with Applications, vol. 40, 2013, pp. 7168-7183.

[14] Q. Lin, Y. Han, and H. Hahn, "Real-Time Lane Departure Detection Based on Extended Edge-Linking Algorithm," Second International Conference on Computer Research and Development, 2010, pp. 725-730.

[15] G. Madey. B. Blake, C. Poellabauer, H. Lu, R. McCune, and Y. Wei, "Applying DDDAS Principles to Command, Control and Mission Planning for UAV Swarms," International Conf. on Computation Science (ICCS), 2010, pp. 1177-1186.

[16] N. Mansurov, (July 7, 2014) "Why Downsampling an Image Reduces Noise," Available: https://photographylife.com/why-downsampling-an-image-reduces-noise.

[17] L. Peng, D. Lipinski, and K. Mohseni, "Dynamic Data Driven Application System for Plume Estimation using UAVs," Journal of Intelligent & Robotic Systems, vol. 74, 2013, pp. 421-436.

[18] R. Lysecky, N. Sandoval, S. Whitsitt, C. Mackin, and J. Sprinkle, "Efficient Reconfiguration Methods to Enable Rapid Deployment of Runtime Reconfigurable Systems," Asilomar Conf. on Signals, Systems and Computers, 2013.

[19] S. Neema, T. Bapty, J. Scott, and B. Eames, 2005. "Signal processing platform: a tool chain for designing high performance signal processing applications," In Proc. of IEEE SoutheastCon, 2005, pp. 302- 307.

[20] S. Neema, J. Sztipanovits, G. Karsai, and K. Butts "Constraint-Based Design-Space Exploration and Model Synthesis," Embedded Software, LNCS, pp. 290-305.

[21] W. Pananurak, S. Thanok, and M. Parnichkin, "Adaptive Cruise Control for an Intelligent Vehicle," IEEE International Conference on Robotics and Biometrics (ROBIO), 2008, pp. 1794-1799.

[22] D. H. Parks, and S. S. Fels, 2008. "Evaluation of Background Subtraction Algorithms with Post-processing," IEEE 5th International Conf. on Advanced Video and Signal Based Surveillance (AVSS), 2008, pp. 192-199.

[23] A. Patra, M. Bursik, J. Dehn, M. Jones, M. Pavolonis, E. B. Pitman, T. Singh, P. Singla, and P. Webley, "A DDDAS Framework for Volcanic Ash Propagation and Hazard Analysis," Proceedings of the International Conference on Computational Science (ICCS), vol. 9, 2012, pp. 1090-1099.

[24] A. Yilmaz, O. Javed, and M. Shah, "Object Tracking: A Survey," ACM Computing Surveys, vol. 38, no. 4, art.13, 2006.

[25] B. Lamiroy, and T. Sun, "Precision and Recall Without Ground Truth," 9th IAPR International Workshop on Graphics RECognition (GREC), 2011.

[26] R. R. McCune, and G. R. Madey, "Swarm Control of UAVs for Cooperative Hunting with DDDAS," Proceedings of the International Conference on Computer Science (ICCS), vol. 18, 2013, pp. 2537-2544.

[27] O. Miksik, and K. Mikolajczyk, 2012. "Evaluation of Local Detectors and Descriptors for Fast Feature Matching," *21st International Conf. on Patter Recognition (ICPR),* 2012, pp. 2681-2684.

[28] S. Russell, and P. Norvig, "Beyond Classical Search," in Artificial Intelligence A Modern Approach, 3rd ed. Upper Saddle River: Prentice Hall, 2010, pp. 125-130.

[29] N. Sandoval, S. Mackin, S. Whitsitt, R. Lysecky, and J. Sprinkle, 2013. "Runtime Hardware/Software Task Transition Scheduling for Data-Adaptable Embedded Systems," International Conference on Field-Programmable Technology (FPT), 2013, pp. 342-345.

[30] Vanderbilt University, "GME 5 user's manual," 2005. [Online]. Available: http://www.isis.vanderbilt.edu/sites/default/files/GMEUMan.pdf

[31] A. Vodacek, J. P. Kerekes, and M. J. Hoffman, "Adaptive Optical Sensing in an Object Tracking DDDAS," Proceedings of the International Conference on Computational Science (ICCS), vol. 9, 2012, pp. 1159-1166.

[32] S. Zhou, Y. Jian, J. Xi, J. Gong, G. Xiong, and H. Chen, "A Novel Lane Detection based on Geometrical Model and Gabor Filter," IEEE Intelligent Vehicles Symposium (IV), 2010, pp. 59-64.

Model-based Fuzzy Logic Classifier Synthesis for Optimization of Data-Adaptable Embedded Systems

Adrian Lizarraga, Roman Lysecky, Jonathan Sprinkle

Dept. of Electrical and Computer Engineering
University of Arizona
Tucson, AZ, USA
adrianlm@email.arizona.edu, rlysecky@ece.arizona.edu, sprinkle@ece.arizona.edu

ABSTRACT

Dynamic data-driven applications such as object tracking, surveillance, and other sensing and decision applications are largely dependent on the characteristics of the data streams on which they operate. The underlying models and algorithms of data-driven applications must continually adapt at runtime to changes in data quality and availability in order to meet both functional and designer-specified performance requirements. Given the dynamic nature of these applications, point solutions produced by traditional design tools cannot be expected to perform adequately across varying execution scenarios. Additionally, the increasing diversity and interdependence of application requirements complicates the design and optimization process. To assist designers of data-drive applications, we present a modeling and optimization framework that enables developers to model an application's data sources, tasks, and exchanged data tokens; specify application requirements through high-level design metrics and fuzzy logic based optimization; and define an estimation framework to optimize the application at runtime. We demonstrate the modeling and optimization process via an example application for video-based vehicle tracking and collision avoidance. We analyze the benefits of runtime optimization by comparing the performance of static point solutions to dynamic solutions over five distinct execution scenarios, showing improvements of up to 74% for dynamic over static configurations. Further, we show the benefits of using fuzzy logic based functions over traditional weighted functions for the specification and evaluation of competing high-level metrics in optimization.

Keywords

Software modeling; dynamic data-driven systems; dynamic optimization; design space exploration; fuzzy logic

1. INTRODUCTION

An increasing number of applications executing on distributed embedded systems operate on vast dynamic data streams typically consisting of video, audio, or other sensory information. Examples of such applications, typically known as dynamic data-adaptable application systems (DDDAS), include structural health monitoring [4], video-based object tracking [6][7][16], and flight planning of unmanned aerial vehicles (UAV) [1][10] among many others. The defining characteristic of such applications is adaptability in response to changing data qualities, data availability, operational modes, and constraints. Specifically, the underlying task implementations and models should dynamically adapt to the emergent characteristics of data in order to preserve functionality, compensate for degradation, or meet performance goals. For example, a detected degradation in received audio quality could be offset by more aggressive filtering or processing algorithms.

Determining the appropriate system adaptation in response to changes in data characteristics or the application's operational state is a complex task that requires a thorough understanding of the relationship between an application's tunable parameters, such as algorithm or hardware configurations, and the high-level metrics representing the performance goals. Further complicating matters, high-level metrics, such as latency and energy consumption, are typically competing such that improving one metric may negatively impact another. The increasing complexity and interdependence of these high-level metrics and tunable parameters accordingly increase the difficulties associated with the design and optimization of these Data-Adaptable Embedded Systems (DAES).

To enable both the design and synthesis of runtime DAES applications in the face of competing optimization metrics, we introduce an extension to the Data-Adaptable System Model (DASM) tool [1]. The DASM modeling environment enables designers of DAES to: 1) model an application's task flow; 2) model the data types consumed and produced by tasks; and 3) specify alternative task implementations along with their input data requirements and models for estimating output data qualities. The proposed extensions to DASM, which constitute the main contributions of this paper, enable designers to: 1) specify the competing high-level metrics to be optimized; 2) specify fuzzy-based fitness rules that capture the relative importance of each high-level metric in determining overall system fitness; and 3) model the transforms that estimate fuzzy classifications for each high-level metric at runtime. Using the designer-specified models, DASM generates a prototype version of the runtime estimation and optimization framework that determines an optimal, or near-optimal, task implementation configuration given the specified fuzzy logic based optimization criteria.

The paper is organized as follows. Section 2 describes related work in the modeling and optimization of embedded systems. Section 3 summarizes the basic modeling capabilities of the DASM modeling environment. Sections 4 introduces the fuzzy logic based system fitness specification framework used in estimation and optimization, which constitutes the main contribution of this paper. Section 5 summarizes the extensions to the runtime optimization algorithms. Section 6 presents the experimental setup and results, and finally Section 7 concludes and discusses future work.

2. RELATED WORK

Model-based techniques for the optimization of embedded systems have been employed by previous research works. The MILAN [3] framework supports the design, simulation, and synthesis of system on chip (SoC) applications using model-based techniques. Using a signal flow modeling paradigm, MILAN enables designers to model application tasks, resources, and constraints. For each task, the modeler specifies alternate implementations (e.g., C or VHDL) for each compatible resource

along with a known latency or energy cost associated with the implementation. Using a human-in-the-loop process, the designer may simulate the application model to verify functionality and calculate better approximations for the latency and energy costs associated with each task implementation.

The Signal Processing Platform (SPP) tool-suite [12] provides a similar modeling language called Signal Processing Modeling Language (SPML) that is specialized for signal processing applications. However, unlike MILAN, the SPML supports parameterized tasks instead of alternative task implementations. Both MILAN and SPP utilize the DESERT [13] tool for design space exploration. DESERT prunes the design space, producing a set of task-to-resource mappings that meet resource and performance (i.e., latency and energy) constraints.

The SPP and MILAN tools are design time tools that utilize known performance characteristics to optimize system configurations. DAES applications, however, must adapt to dynamic data characteristics and emergent constraints at runtime, thus requiring new tools and methodologies. Additionally, designers often need to optimize DAES applications with respect to various application-specific high-level metrics, not just latency and energy. Further, optimization tools must also support a framework for the specification of competing high-level metrics.

The DARES project [9][15] models data configurability of application tasks supporting the runtime reconfiguration of hardware accelerators implemented with an FPGA. Unlike other approaches, the application model can be synthesized into a physical device in order to enable runtime task allocation in order to optimize performance. However, the DARES approach does not consider the specification or adaptability of the algorithms themselves, only the type of data being processed. Additionally, DARES does not support the specification of competing high-level metrics as the target for optimization.

3. DATA-ADAPTABLE SYSTEM MODELING

3.1 Case Study: Video-based Vehicle Tracking and Collision Avoidance Application

We utilize a video-based vehicle tracking and collision avoidance (VBVTCA) application throughout this paper in order to demonstrate DASM's modeling and runtime optimization capabilities. The VBVTCA application executes as a part of an autonomous vehicle's distributed embedded system network with the purpose of avoiding collisions with vehicles traveling in the same lane as depicted in Figure 1.

Using video from a vehicle-mounted video camera, the VBVTCA application analyzes incoming video frames to detect and track vehicles, hereafter referred to as *lead vehicles*, in the same lane of travel. Once a lead vehicle has been detected and tracked, the application estimates the lead vehicle's relative position, speed, and acceleration. By comparing the lead vehicle's positional data to the autonomous vehicle's own positional data, the application calculates the necessary braking force and the resulting minimum traveling distance (MTD) necessary to match the lead vehicle's speed and avoid a rear-end collision.

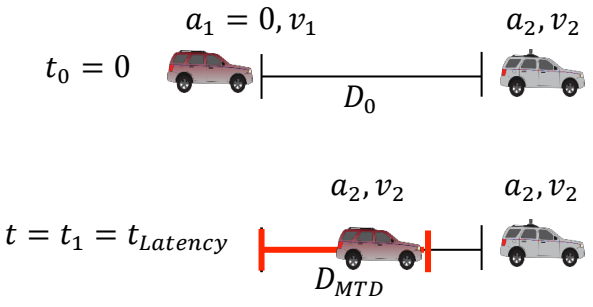

Figure 1. Illustration demonstrating collision avoidance by slowing down and matching lead vehicle's speed.

The VBVTCA application is a suitable case study for demonstrating the DASM methodology due to the inherent variability in video data characteristics (e.g., resolution, signal-to-noise ratio), and the large number of potential object detection and tracking task implementation alternatives [6][16][7], each suited for inputs of certain characteristics.

3.2 Modeling Tasks, Task Options, and Data Types

The DASM modeling environment enables DDDAS designers to model the task flow of an application, as shown in Figure 2. A task flow is a set of connected tasks that defines the functionality of an application. Tasks represent abstract executable computations that consume and produce data. The model depicted in Figure 2 for the VBVTCA application contains ten tasks, which are now described.

The Video task captures video frames from a video source. The Downsampling tasks, DS1 and DS2, optionally lower the resolution of input video frames. The Background Subtraction (BS) task extracts moving objects in an image to produce a foreground mask. The Morphological Filter (MF) task filters small objects from the foreground mask and uses connected components labeling to uniquely identify different objects in the image. The Feature Detection/Tracking task analyses object features to track objects between frames. The Inverse Perspective Mapping (IPM) task uses known camera and geometric parameters to estimate the lead vehicle's distance, i.e., position, and the Position (P), Velocity (V), and Acceleration (A) tasks estimate the lead vehicle's kinematic attributes. The Minimum Travel Distance Calculation (MTDC) task calculates the MTD necessary to match the lead vehicle's speed.

A task must specify one or more alternative implementations known as Task Options. Task Options may represent different algorithms for a computational task or different models for simulation tasks. The model in Figure 2, for example, shows the three available task options for the BS task: Gaussian Mix, Adaptive, and 1 Gaussian.

A Data Type, which is depicted as a rectangle with a folded corner in Figure 2, represents a data token transferred between tasks. The model in Figure 2 shows two data types, namely the Downsampled Video data type, which is produced by the DS1 task and consumed by the BS task, and the Foreground Mask data type, which is produced by the BS task and consumed by the DS2 task. All task options within a certain task must consume and produce the same data types in order to ensure compatibility with neighboring tasks.

Data types contain two types of attributes. Data attributes (DA) are properties, such as resolution, that describe a data type and represent the assumptions made on the data inputs and outputs of a task. Evaluation attributes (EA) describe the quality of data types and are used to evaluate task options during optimization. For example, the latency EA represents the execution latency associated with the task option that produced the data type, and the accuracy EA represents the performance (e.g., precision or recall) of algorithms such as background subtraction or feature detection.

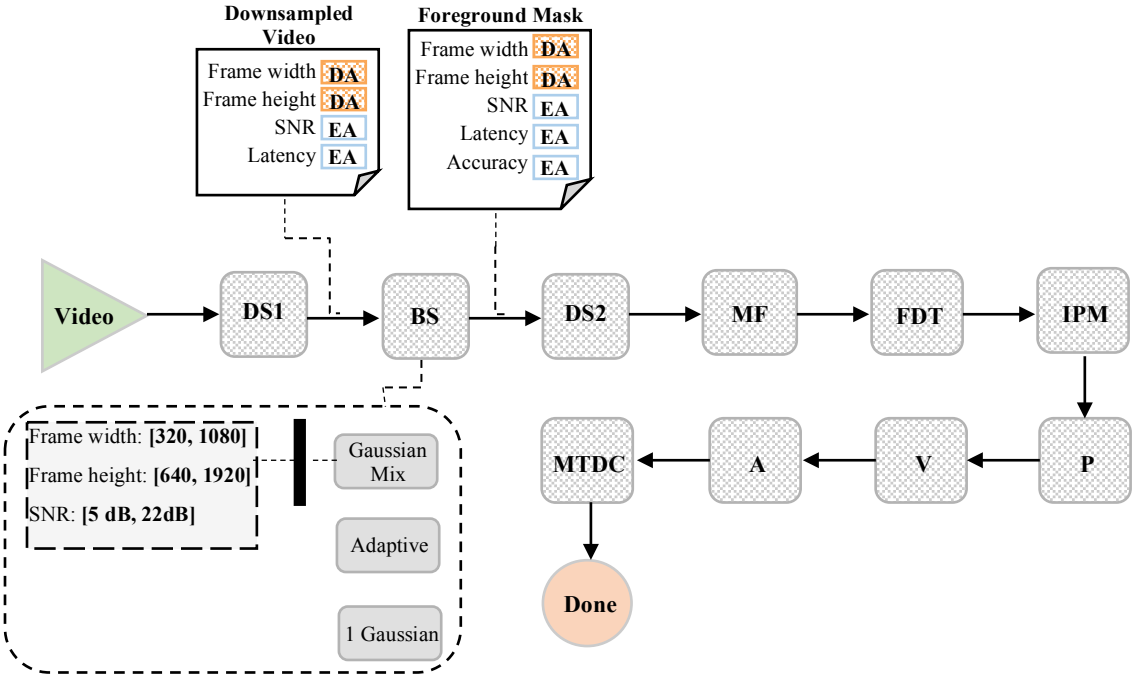

Figure 2. Task flow modeling for a video-based vehicle tracking and collision avoidance application. Modeling a data-adaptable application requires specifying the end-to-end task flow, data types, and available task options.

3.3 Modeling Attribute Guards

For each task option, designers may specify an Attribute Guard to define or restrict the type of input data types for which the task option is valid at runtime. For example, the black vertical bar in Figure 2 represents an attribute guard specifying that the Gaussian Mix task option is only valid for Downsampled Video data types with a resolution ranging from 320x640 to 1080x1920 and a signal-to-noise ratio (SNR) ranging from 5dB to 22dB.

Attribute guards may specify the semantic and programmatic composability of task options. Semantic composability defines the range of input values for which the task option is known to be functional. Programmatic compatibility captures designer knowledge on the suitability of a task option for a certain range of input values. For example, although the Gaussian Mixtures task option in Figure 2 can process input video with a much larger range of SNR values, the attribute guard restricts the Gaussian Mixtures task option to inputs between 5dB and 22dB because prior design experience indicates that the Gaussian Mixtures algorithm only exhibits appreciable performance (e.g., execution latency and accuracy) gains over less complex implementations (e.g., adaptive background subtraction) when operating on data with lower SNR values [5][7][14].

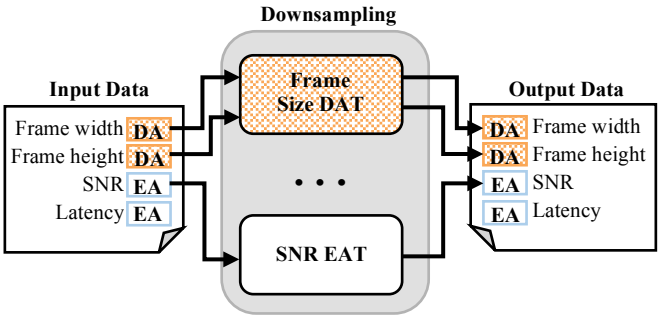

Figure 3. A Task Option transforms data attributes using Data Attribute Transforms and Evaluation Attribute Transforms.

3.4 Modeling Data and Evaluation Attribute Transforms

A task option is an abstraction of an actual algorithm that transforms an input data type (e.g., Video Frame) to an output data type (e.g., Downsampled Frame). A designer defines the semantics of such data type transformations by using attribute transforms. As shown in Figure 3, a Data Attribute Transform (DAT) takes in input data attributes and generates output data attributes. Similarly, an Evaluation Attribute Transform (EAT) takes in input evaluation attributes and generates output evaluation attributes.

Attribute transforms are not necessarily actual implementations of a task option algorithm, but instead are often fast approximations implemented as mathematical expressions, simple routines, or even look-up tables.

3.5 Modeling Computational Resources and Communication Delay

The value of various evaluation attributes, such as latency, may depend on the characteristics of the computational resource on which a task executes. Therefore, DASM enables designers to model basic computational resources consisting of processing, memory, and communication elements as shown in Figure 4.

A designer specifies instruction set architecture characteristics such as cycles per integer operation for each processing element (PE) in a device. For memory elements, a designer specifies properties such as cache associativity, cache size, etc. The Intra-PE Communication elements model communication delay between PEs in the same device (e.g., PE_0 and PE_1), and the Inter-PE Communication elements model communication delay between PEs in different devices. These

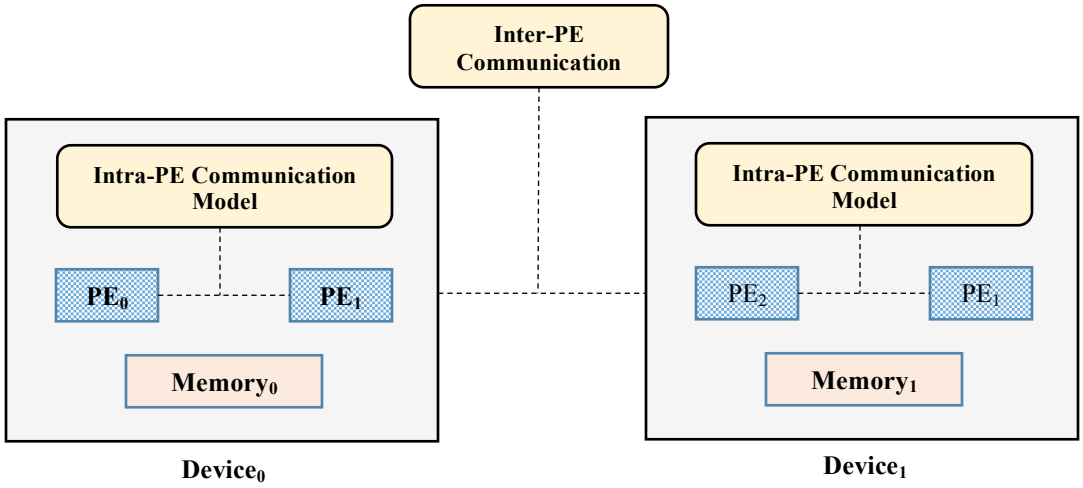

Figure 4. Example of a designer-specified Resource and Communication model within DASM showing two devices, each with two processing elements, a memory and intra-PE communication models. The inter-PE communication block models communication between PEs on separate devices.

communication models are implemented by designers as EATs that calculate latencies from input data sizes (e.g., video resolution).

At design time, each task option is associated with a base PE from which its latency EAT was derived. At runtime, the optimization framework dynamically maps tasks to various different devices and scales the task option's resulting latency EA by a scaling factor derived from the base and newly mapped devices. This scaling factor denotes the relative speedup or slow-down of the new task-to-device mapping and is a function of the PE characteristics, memory characteristics, and task option benchmarks (e.g., average CPI, average miss rate).

4. FUZZY LOGIC BASED SYSTEM FITNESS SPECIFICATION

4.1 Fuzzy Design Metric Classification

The DASM modeling environment allows developers of DAES applications to model application-specific data types transferred throughout the various tasks. Any of the evaluation attributes associated with these data types, including the application's end-to-end latency, can be denoted as *high-level metrics* by the designer for optimization. For the VBVTCA application in Figure 2, the accuracy and end-to-end latency evaluation attributes associated with the data type output of the MTDC task are denoted as high-level metrics. Note that the accuracy metric represents the confidence level, as a percentage, of the application's MTD calculation.

The DASM modeling and optimization frameworks use a fuzzy logic based formalism to interpret the fitness of individual high-level metrics and the overall system performance. In comparison to other formalisms, such as weighted piecewise linear functions, fuzzy logic allows designers to more intuitively specify tradeoffs between competing metrics and optimization goals. In fact, other research [8][11] has demonstrated that fuzzy logic formalisms can achieve greater results than piecewise linear functions.

Converting a raw high-level metric value, such as a latency of 100ms, to a fuzzy classification requires the specification of fuzzy metric classification functions. Figure 5(a) presents two example fuzzy metric classification functions that could be utilized for the VBVTCA application for the classification of the latency and accuracy high-level metrics.

A fuzzy metric classification function maps a raw metric value to one of four discrete classifications. In increasing order of fitness, the four discrete classifications include *Unacceptable,*

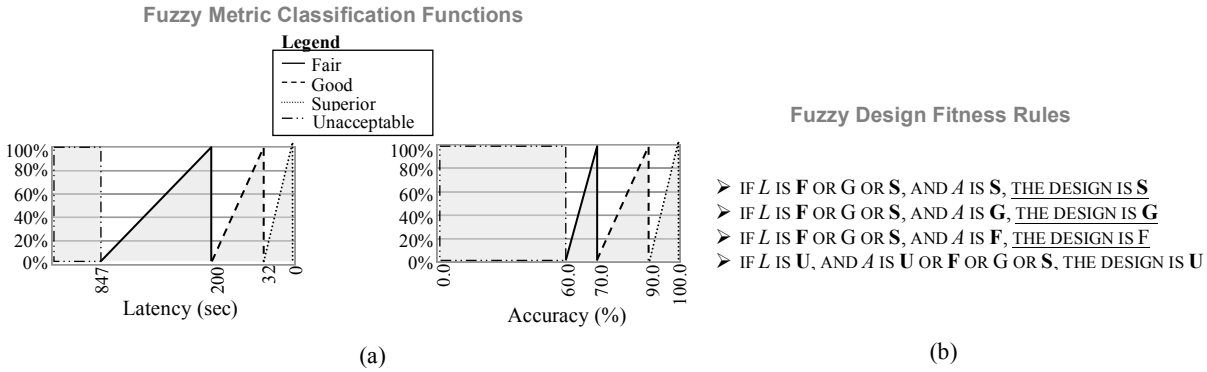

Figure 5. Defining models for the fitness estimation of a task option configuration involves (a) specifying fuzzy metric classification functions that relate a raw metric value to a fuzzy classification, and (b) specifying fuzzy design fitness rules that relate the relative importance of each fuzzy high-level metric in determining the overall system fitness.

Fair, Good, and *Superior*. For example, the latency classification function in Figure 5(a) indicates that latencies between 0ms and 32ms are *Superior*, latencies between 32ms and 200ms are *Good*, latencies between 200ms and 847ms are only *Fair*, and latencies greater than 847ms are *Unacceptable*. Additionally, the y-axis of fuzzy metric classification functions map a raw metric value to a percent membership within a certain classification. The percent membership value indicates how *Unacceptable, Fair, Good,* or *Superior* a metric is. For example, Figure 5(a) indicates that a latency of 16ms is 50% *Superior* and that an accuracy of 80% is 50% *Good*.

Note that the percent membership of a metric within a discrete fuzzy classification is defined using a linear function. Additionally, metric values that extend past the specified *Superior* classification are mapped to 100% *Superior*. That is, percent membership values cannot be greater than 100%.

4.2 Fuzzy Design Fitness Rules

To specify the relative importance of each high-level metric in determining the overall system fitness, designers are tasked with specifying fuzzy design fitness rules within the DASM modeling environment. Fuzzy design fitness rules are English sentences that map the fuzzy classification of each metric to a single fuzzy classification that defines the fitness of the overall system.

For example, Figure 5(b) presents four fuzzy design fitness rules for the VBVTCA application. The first rule specifies that if the latency is *Fair*, *Good*, or *Superior*, and the accuracy is *Superior*, then the overall system fitness is *Superior*. Similarly, the last rule indicates that if latency is *Unacceptable*, the overall system fitness is also *Unacceptable* regardless of the achieved accuracy. The designer must minimally define a set of fuzzy design fitness rule that specify at least one *Superior*, *Good*, *Fair*, and *Unacceptable* design. Note that if a two or more fuzzy design fitness rules overlap such that the same set of metric classifications map to multiple fuzzy design fitness rules, only the fuzzy design fitness rule resulting in the best system quality is utilized.

Designs mapped to an overall system fitness classification must be associated with a percent membership value to enable comparisons between designs. DASM calculates the percent membership value of a design within a certain fuzzy design fitness rule *i*, which is denoted as $\%MembershipFFR^i$, according to equation (1). As an example, a $\%MembershipFFR^1$ value of 40% for the first fuzzy fitness rule in Figure 5(b) indicates that a design has an overall fitness of 40% *Superior*.

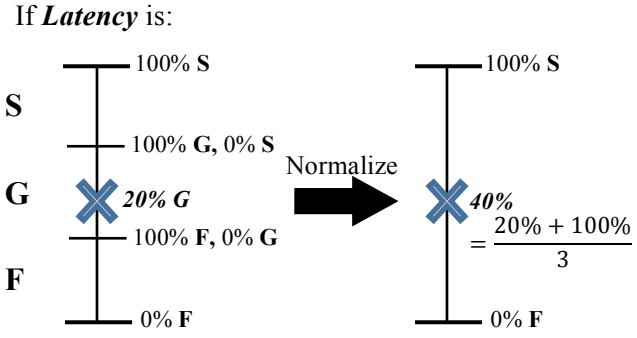

Figure 6. Example of fuzzy metric percent membership normalization for a fuzzy design fitness rule that requires a latency that is *Good, Fair*, or *Superior*. A latency membership value of 20% G is normalized to 40% for that rule.

$$\%MembershipFFR^i = \sum_{m=1}^{NumMetrics} \frac{\%Membership_{NORM}^{m,i}}{NumMetrics} \quad (1)$$

Equation (1) calculates $\%MembershipFFR^i$ of a set of fuzzy high-level metrics for a specific fuzzy design fitness rule *i* by averaging the *normalized* percent membership values, $\%Membership_{NORM}^{m,i}$, of each high-level metric, where the superscript *m* identifies a specific high-level metric. As shown in equation (2), the $\%Membership_{NORM}^{m,i}$ value for each metric is a function of the metric's percent membership value within its fuzzy metric classification, i.e., $\%Membership^m$, and the metric's classification requirements within the fuzzy design fitness rule *i*.

$$\%Membership_{NORM}^{m,i} = \frac{\%Membership^m + Offset^{m,i}}{NumClassifications^{m,i}} \quad (2)$$

The $NumClassifications^{m,i}$ value represents the number of classifications for metric *m* in fuzzy design fitness rule *i.* For example, the $NumClassifications$ value for the latency metric for the first fuzzy design fitness rule in Figure 5(b) is 3. The value $Offset^{m,i}$ is a numerical representation of the metric *m*'s classification.

To better illustrate the metric normalization process of equation (2), Figure 6 illustrates the normalization of a latency metric that evaluates to 20% *Good* according to its fuzzy metric classification function. Considering the first fuzzy design fitness rule in Figure 5(b), the $NumClassifications$ value is 3 and the *Offset* for latency is 100%. This results in a normalized latency value of 40%.

In summary, DASM interprets a set of fuzzy metric classifications, such as a latency and accuracy that are 20% *Good* and 50% *Superior* respectively, to an overall design fitness such as 45% *Superior*.

4.3 Fuzzy Logic Classifier Synthesis

Fuzzy metric classification functions define the "goodness" of high-level metrics by mapping a raw metric value to a fuzzy classification. However, given the dynamic nature of DAES applications, a single static fuzzy metric classification function may fail to accurately define the goodness of a certain high-level metric across all dynamic execution scenarios.

For example, in the VBVTCA application the meaning of what constitutes a *Fair* or *Unacceptable* end-to-end latency depends heavily on the relative speeds of nearby vehicles. Specifically, the VBVTCA application must react much faster in a highway setting than in school-zone, where the expected vehicle

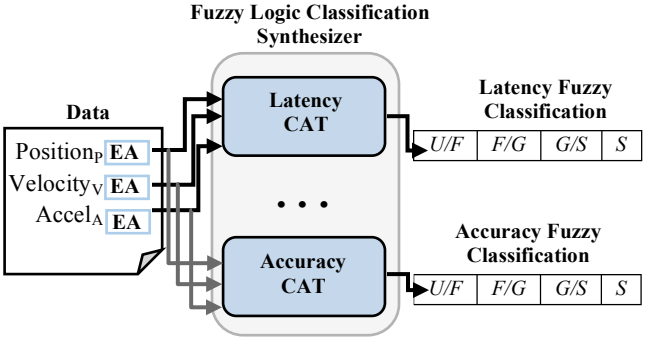

Figure 7. A Fuzzy Logic Classification Synthesizer model consists of classifier attribute transforms, which synthesize a metric's fuzzy metric classification function at runtime based on the runtime values of evaluation attributes.

speeds are much slower. Therefore, while an application latency value of 750ms may be classified as *Fair* in a residential area, it may be *Unacceptable* in a highway scenario where waiting 750ms between MTD calculations equates to more than 20 meters of uninformed, i.e., "blind", driving.

In general, dynamically changing data qualities and execution scenarios demand adaptable fuzzy metric classifications. Therefore, DASM supports runtime synthesis of fuzzy metric classification functions via the specification of a Fuzzy Logic Classification Synthesizer (FLCS) model at design time.

A FLCS model, as shown in Figure 7, consists of several Classifier Attribute Transforms (CAT), which transform data and evaluation attributes into a numerical description of a fuzzy metric classification function. Specifically, a designer is tasked with specifying a CAT for each high-level metric of interest. Similar to EATs and DATs, a designer may specify a CAT as a simple mathematical equation, a look-up table, or as a generic function. Note that each CAT produces four numbers, which correspond to the classification boundaries in a fuzzy metric classification function. For example, the *U/F* value for latency in Figure 7 corresponds to the *Unacceptable-Fair* classification boundary for latency. The four classification boundary values are sufficient to reconstruct a fuzzy metric classification function.

For the VBVTCA application, the FLCS model contains CATs for the latency and accuracy metrics. These CATs utilize the EAs from the Position, Velocity, and Acceleration tasks in order to synthesize fuzzy metric classification functions at runtime based on the relative kinematic attributes of the vehicles. The latency CAT determines the latency classification boundaries by first calculating the maximum tolerable latency, $Latency_{max}$, necessary to safely calculate the MTD, and then assigning this maximum latency value to the *U/F* boundary. The $Latency_{max}$ value is derived by using a simple kinematic model of the autonomous and lead vehicles, in which $Latency_{max}$ corresponds to the maximum application latency necessary to just barely match the lead vehicle's speed as the two vehicles come into contact.

The other classification boundaries for latency were determined using a look-up table. Similarly, the Accuracy CAT determines the accuracy fuzzy metric classification boundaries using a look-up table indexed with the relative speed of the lead vehicle. Although, more sophisticated models can be used in a physical implementation, the aforementioned CAT models suitably convey the utility and functionality of the FLCS model.

5. RUNTIME MODEL-GUIDED OPTIMIZATION

Based on the designer-specified models for an application's task flow, task options, data types, and fuzzy fitness specifications, the DASM tool currently supports the generation of a prototype runtime optimization framework, which is presented in Figure 8 for the VBVTCA application.

The DASM runtime optimization framework maintains a local model of the application's task flow in order to dynamically estimate evaluation attributes. At runtime, the fuzzy metric classification functions for all high-level metrics are synthesized from state information consisting of dynamically determined evaluation attribute values. The DASM DSE algorithms use the resulting fuzzy metric classifications functions to search the design space for an optimal or near-optimal system configuration of task options, T_1 through T_N, and task-to-PE mappings, M_1 through M_N, as specified by (3) and where N indicates the number of tasks.

$$Config = < T_1, T_2, \ldots, T_N, M_1, M_2, M_N > \quad (3)$$

Notably, the DSE phase utilizes a model-guided genetic algorithm (GA), which is described in more detail in [1], to explore and evaluate the design space. The model-guided genetic algorithm achieves performance improvements over a standard genetic algorithm by utilizing the information from the designer-specified models to aggressively prune the design space.

In the previous implementation of the model-guided GA, the fitness function for the VBVTCA application was a simple piecewise linear function that used weights to define the relative importance of the latency and accuracy high-level metrics. Additionally, the raw end-to-end latency metric, which incorporates device scaling and scheduling, was transformed using simple a linear function where a raw latency of $Latency_{max}$ mapped to a value of zero and a raw latency of 30ms mapped to a value of 100.

For the DASM extension presented in this paper, the model-guided GA was revised to use the aforementioned fuzzy logic based formalisms for the evaluation of configurations.

6. EXPERIMENTAL RESULTS

In order to evaluate the benefits of model-guided fuzzy logic

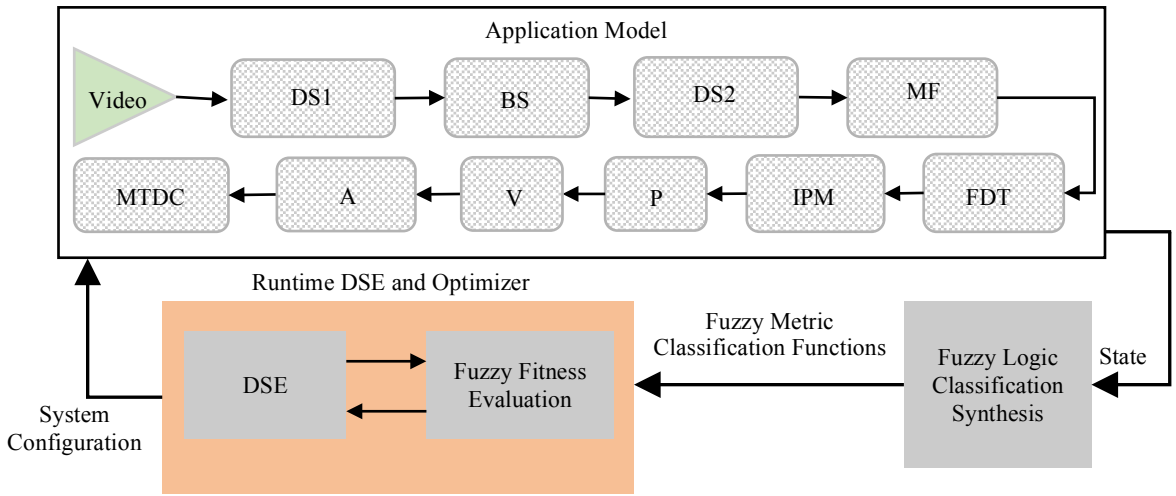

Figure 8. Prototype of the DASM runtime optimization framework for the VBVTCA application.

Table 1. Synthesized fuzzy metric classification functions for latency and accuracy over five execution scenarios.

Scenario	ΔSpeed (mph)	Latency Fuzzy Classification Boundaries				Accuracy Fuzzy Classification Boundaries			
		U/F	F/G	G/S	S	U/F	F/G	G/S	S
1	4	847	200	32	0	60%	70%	90%	100%
2	8	749	200	32	0	50%	60%	85%	100%
3	14	619	200	32	0	50%	60%	80%	100%
4	16	508	200	32	0	50%	60%	75%	100%
5	23	235	200	32	0	50%	60%	70%	100%

classification synthesis for the optimization of DAES applications, we executed the prototype runtime optimization framework for the VBVTCA application and selected the five dynamic execution scenarios, which are presented in Table 1, for further analysis.

Each of the five execution scenarios is characterized by the detected speed differential between the autonomous and lead vehicles. The first scenario, with a Δspeed of 4 mph, represents a school-zone, while the fifth scenario, with a Δspeed of 23 mph, could potentially represent a highway. As discussed in Section 4.3, the FLCS model dynamically generates fuzzy metric classification functions based on the runtime value of the EAs and the designer-specified CAT models. Table 1 additionally lists the synthesized metric classification functions for each execution scenario using the fuzzy design fitness rules specified in Figure 5.

First, we analyze the benefits of utilizing the presented fuzzy logic based formalisms during DSE by comparing to the results obtained using piecewise linear weighted functions, which are described in Section 5. For each execution scenario, Figure 9 compares the fitness of optimal configurations obtained using fuzzy logic based formalisms to the fitness of optimal configurations obtained using a piecewise weighted linear function. For scenarios 1 thorough 4, the configurations obtained using fuzzy logic exhibit a moderate degradation in latency fitness in exchange for significant increases in accuracy and overall system fitness. Note that both methods found the same optimal configuration in scenario 5.

On average, the fuzzy based approach produced configurations that sacrificed latency by 8.4%, but showed average improvements in accuracy and overall system fitness of 25.7% and 67.3%, respectively. Importantly, although the fuzzy

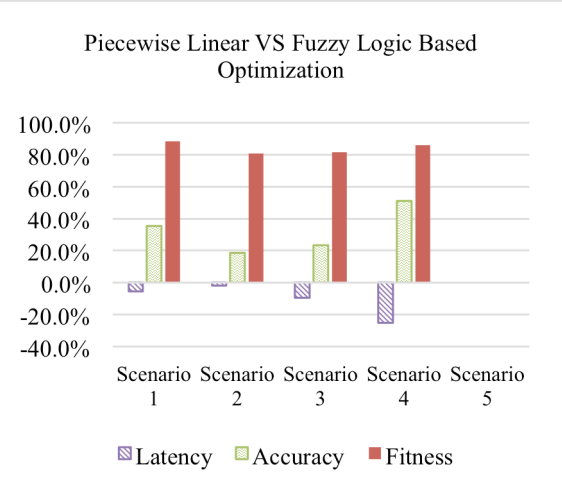

Figure 9. Improvement in metric and overall system fitness due fuzzy logic based optimization in comparison to piecewise weighted linear functions.

based approach produced "slower" configurations, these configurations still executed faster than the limiting value of $Latency_{max}$. The fuzzy based configurations still execute within the application's safety constraints, and additionally achieve significant increases in fitness. Thus, it is evident that the fuzzy logic based formalisms more precisely characterize latency as a constraint that must only be met, while accuracy is aggressively optimized.

To evaluate the benefits of dynamically adapting a system configuration instead of utilizing a single static configuration, we evaluated optimal configurations in scenarios for which these configurations were not optimal. We first determined the optimal configuration in each execution scenario described in Table 1. That is, configuration 1 denotes the optimal configuration for scenario 1, configuration 2 denotes the optimal configuration for scenario 2, and so on. Then, we evaluated the fitness of each configuration in the other scenarios as shown in Figure 10.

For scenario 1, configuration 2 exhibits degradations of 14.3%, 43.5%, and 69.4% in latency, accuracy, and overall fitness, respectively. Because configuration 2 is optimal for an execution scenario with similar fuzzy metric classification functions to those used in scenario 1, configuration 2 does not exhibit as large a degradation as configuration 3, which utilizes more lenient metric classification functions.

Note that configurations 4 and 5 in scenario 1 are shown as invalid due to resolution violations (RV). In order to avoid *Unacceptable* latency classifications in their respective execution scenarios, configurations 4 and 5 utilized an extremely low resolution video to achieve faster latencies. Such low resolutions are deemed unsafe in a school-zone.

In scenario 2, configuration 1 is invalid due to an *Unacceptable* latency violation (LV). In fact, configuration 1's latency, which is just inside scenario 1's *Fair* classification boundary, is actually *Unacceptable* in all other scenarios. Similarly, all non-optimal configurations in scenario 5 are also invalid due to the strictness of the latency classification boundaries, which specify that latencies above 235ms are *Unacceptable*.

In summary, the results in Figure 10 indicate that any single optimal configuration can exhibit unacceptable degradations or even be classified as unacceptable over several execution scenarios. Therefore, a single static point solution is inadequate for DAES applications.

7. CONCLUSIONS AND FUTURE WORK

We presented modeling and optimization extensions to the DASM tool that enable more intuitive tradeoff specifications for competing high-level metrics. Specifically, the DASM modeling environment now supports the designation of multiple, potentially competing, evaluation attributes as high-level metrics targetable for optimization. The Fuzzy Logic Classification Synthesizer model enables runtime synthesis of fuzzy metric classification

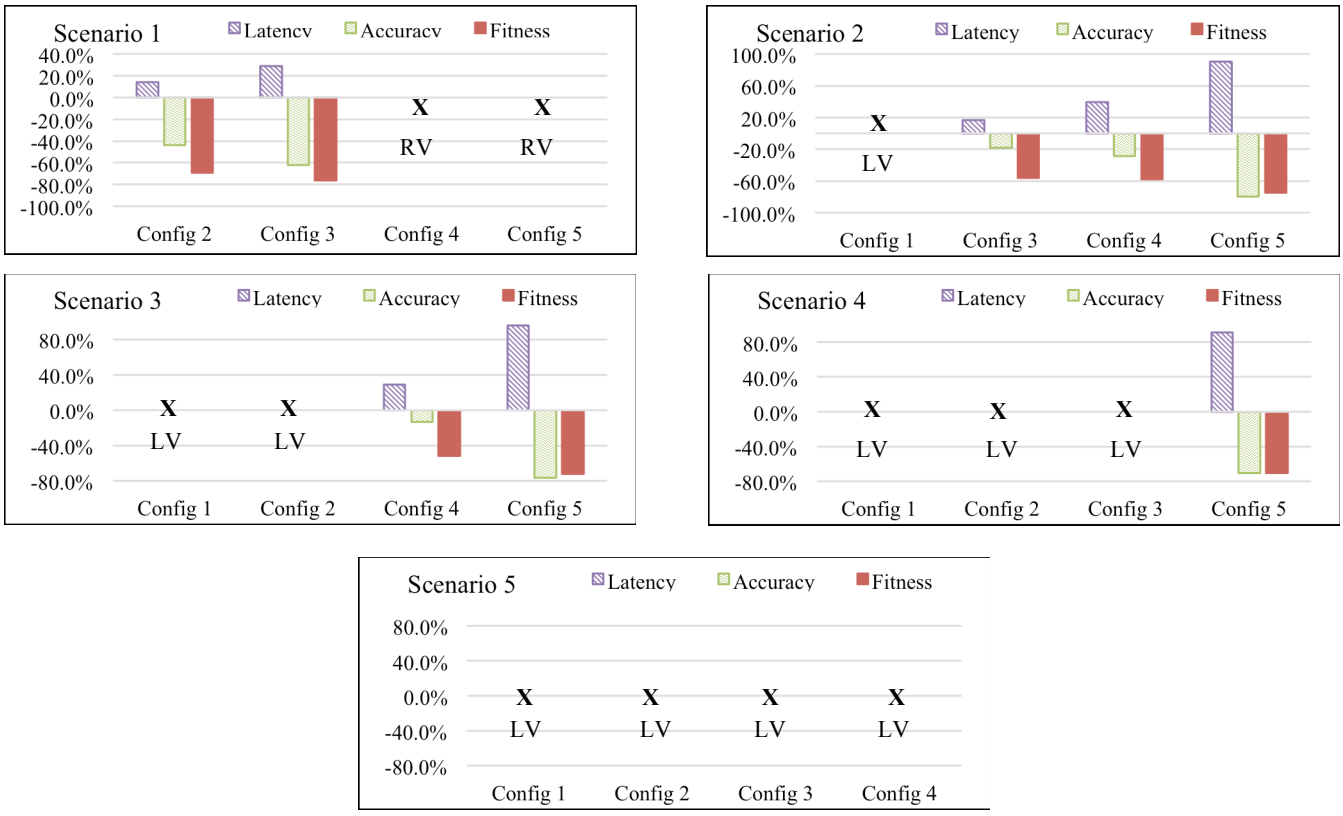

Figure 10. Relative performance of the non-optimal configurations over the five optimization scenarios. Each graph presents an optimization scenario showing the percent deviation in latency, accuracy, and utilization of each configuration in comparison to the optimal configuration. Video resolution violations are indicated with the acronym RV, and *Unacceptable* latency violations are indicated with an LV.

functions, which map raw metric values to a fuzzy quality. Additionally, the DASM modeling environment supports the specification of fuzzy design fitness rules to define the relative importance of competing metrics when determining overall system fitness.

Results indicate that using fuzzy logic based formalisms for optimization – instead of our previous piecewise linear function approach – allows a designer to more intuitively specify fitness models that can produce configurations that more faithfully characterize the intended performance goals. Additionally, results continue to demonstrate that the runtime adaptability provided by the DASM tool is necessary for maintaining both the functionality and optimality of DAES applications.

Future work entails developing a runtime profiling framework that can monitor the actual value of evaluation attributes at runtime in order to train or refine the designer-specified EATs. Additionally, future work should investigate methods for determining the opportunity cost associated with runtime reconfiguration in order to better understand when to optimize. Lastly, future revisions of the DASM should allow designers to specify custom fuzzy classifications.

8. REFERENCES

[1] Withheld for blind review.

[2] Allaire, D., Biros, G., Chambers, J., Ghattas, O., Kordonowy, D., and Willcox, K. 2012. Dynamic Data Driven Methods for Self-aware Aerospace Vehicles. *International Conf. on Computation Science (ICCS)*, 1206-1210.

[3] Bakshi, A., Prasanna, V. K., and Ledeczi, A. 2001. MILAN: A Model Based Integrated Simulation Framework for Design of Embedded Systems. *In Proc. of the 2001 ACM SIGPLAN workshop on Optimization of middleware and distributed systems,* 82-93.

[4] Bazilevs, Y., Marsden, A. L., Lanza di Scalea, F., Majumdar, A., and Tatineni, M. 2012. Toward a Computation Steering Framework for Large-Scale Composite Structures Based on Continually and Dynamically Injected Sensor Data. *International Conf. on Computation Science (ICCS),* pp. 1149-1158.

[5] Benezeth, Y., Jodoin, P., Emile, B., Laurent, H., and Rosenberger, C. 2010. Comparative study of background subtraction algorithms. *Journal of Electronic Imaging, Society of Photo-optical Instrumentation Engineers (SPIE).*

[6] Chintalacheruvu, N., and Muthukumar, V. 2012. Video Based Vehicle Detection and Its Application in Intelligent Transportation Systems. *Journal of Transportation Technologies*, 305-314.

[7] Gedeoglu, Y. 2004. Moving Object Detection Tracking and Classification for Smart Video Surveillance. Master of Science Thesis.

[8] Lizarraga, A., Lysecky, R., Lysecky, S., and Gordon-Ross A. 2013. Dynamic Profiling and Fuzzy Logic Based Optimization of Sensor Networks Platforms. *ACM Transactions on Embedded Computing Systems (TECS),* Vol. 13, No. 3, Article 51, pp. 1-29.

[9] Lysecky, R., Sandoval, N., Whitsitt, S., Mackin, C., Sprinkle, J. 2013. Efficient Reconfiguration Methods to Enable Rapid Deployment of Runtime Reconfigurable Systems. *Asilomar Conf. on Signals, Systems and Computers*.

[10] Madey. G., Blake, B., Poellabauer, C., Lu, H., McCune, R., and Wei, Y. 2012. Applying DDDAS Principles to Command, Control and Mission Planning for UAV Swarms. *International Conf. on Computation Science (ICCS)*, 1177-1186.

[11] Mathelin, M., Perneel, C., Acheroy, M. 1993. Bayesian Estimation vs Fuzzy Logic for Heuristic Reasoning. *IEEE International Conference on Fuzzy Systems*.

[12] Neema, S., Bapty, T., Scott, J., and Eames, B. 2005. Signal processing platform: a tool chain for designing high performance signal processing applications. *In Proc. of IEEE SoutheastCon*, 302- 307.

[13] Neema, S., Sztipanovits, J., Karsai, G., and Butts, K. Constraint-Based Design-Space Exploration and Model Synthesis. *Embedded Software, LNCS*, 290-305.

[14] Parks, D. H., and Fels, S. S. 2008. Evaluation of Background Subtraction Algorithms with Post-processing. *IEEE 5th International Conf. on Advanced Video and Signal Based Surveillance (AVSS),* 192-199.

[15] Sandoval, N., Mackin, S., Whitsitt, S., Lysecky, R., Sprinkle, J. 2013. Runtime Hardware/Software Task Transition Scheduling for Data-Adaptable Embedded Systems. *International Conference on Field-Programmable Technology (FPT),* 342-345.

[16] Yilmaz, A., Javed, O., and Shah, M. 2006. Object Tracking: A Survey. *ACM Computing Surveys*, Vol. 38, No. 4, Art.13.

1.

1. Report Type

Final Report

Primary Contact E-mail

Contact email if there is a problem with the report.

rlysecky@ece.arizona.edu

Primary Contact Phone Number

Contact phone number if there is a problem with the report

520-834-3445

Organization / Institution name

University of Arizona

Grant/Contract Title

The full title of the funded effort.

Data-Adaptable Modeling and Optimization for Runtime Adaptable Systems

Grant/Contract Number

AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550-15-1-0143

Principal Investigator Name

The full name of the principal investigator on the grant or contract.

Roman Lysecky

Program Manager

The AFOSR Program Manager currently assigned to the award

Frederica Darema

Reporting Period Start Date

05/01/2015

Reporting Period End Date

04/30/2016

Abstract

Dynamic data driven application systems (DDDAS) involve complex sensing and decision-making algorithms that operate on vast data streams with dynamic characteristics. As the availability and quality of the sensed data changes, the underlying models and decision algorithms should continually adapt in order to meet desired high-level requirements. Due to the complexity of such dynamic data-driven systems, traditional design time techniques are incapable of producing a solution that remains optimal in the face of dynamically changing data, algorithms, and even availability of computational resources. Additionally, modern approaches to DDDAS design the adaptation laws for dynamic behavior as part of the system itself, thereby resulting in a point solution for that specific application. This research project developed generalized approaches to DDDAS so that the benefits of adaptability can be extended to other applications, without resorting to application-specific point solutions.

Distribution Statement

This is block 12 on the SF298 form.

Distribution A - Approved for Public Release

Explanation for Distribution Statement

If this is not approved for public release, please provide a short explanation. E.g., contains proprietary information.

SF298 Form

Please attach your SF298 form. A blank SF298 can be found here. Please do not password protect or secure the PDF The maximum file size for an SF298 is 50MB.

SF298 Lysecky 060216.pdf

Upload the Report Document. File must be a PDF. Please do not password protect or secure the PDF . The maximum file size for the Report Document is 50MB.

dddas_lysecky_final_report_053116.pdf

Upload a Report Document, if any. The maximum file size for the Report Document is 50MB.

Archival Publications (published) during reporting period:

A. Lizarraga, J. Sprinkle, R. Lysecky. Model-driven Optimization of Data-Adaptable Embedded Systems. IEEE Computer Software and Applications Conference (COMPSAC), 2016.

A. Lizarraga, J. Sprinkle, R. Lysecky. Model-based Fuzzy Logic Classifier Synthesis for Optimization of Data-Adaptable Embedded Systems. Submitted to ACM SIGBED International Conference on Embedded Software (EMSOFT), Under Review, 2016.

2. New discoveries, inventions, or patent disclosures:
Do you have any discoveries, inventions, or patent disclosures to report for this period?

No

Please describe and include any notable dates

Do you plan to pursue a claim for personal or organizational intellectual property?

Changes in research objectives (if any):

Change in AFOSR Program Manager, if any:

Extensions granted or milestones slipped, if any:

AFOSR LRIR Number

LRIR Title

Reporting Period

Laboratory Task Manager

Program Officer

Research Objectives

Technical Summary

Funding Summary by Cost Category (by FY, $K)

	Starting FY	FY+1	FY+2
Salary			
Equipment/Facilities			
Supplies			
Total			

Report Document

Report Document - Text Analysis

Report Document - Text Analysis

Appendix Documents

2. Thank You

E-mail user

Jun 07, 2016 12:12:37 Success: Email Sent to: rlysecky@ece.arizona.edu